Basic DTU Wind Energy controller

Hansen, Morten Hartvig; Henriksen, Lars Christian

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

DTU Wind Energy

Basic DTU Wind Energy controller

Department of Wind Energy Report 2013

Morten Hartvig Hansen and Lars Christian Henriksen

DTU Wind Energy E-0028

January 2013

DTU Vindenergi
Institut for Vindenergi

Author(s): Morten Hartvig Hansen and Lars Christian Henriksen
Title: Basic DTU Wind Energy controller
Institute: Department of Wind Energy

Summary (max. 2000 char.):
This report contains a description and documentation, including source code, of the basic DTU Wind Energy controller applicable for pitch-regulated, variable speed wind turbines. The controller features both partial and full load operation capabilities as well as switching mechanisms ensuring smooth switching between the two modes of operation.
The partial and full load controllers are both based on classical proportional-integral control theory as well as additional filters such as an optional drive train damper and a notch filter mitigating the influence of rotor speed dependent variations in the feedback. The controller relies on generator speed as the primary feedback sensor. Additionally, the reference generator power is used as a feedback term to smoothen the switching between partial and full load operation. Optionally, a low-pass filtered wind speed measurement can be used for wind speed dependent minimum blade pitch in partial load operation.
The controller uses the collective blade pitch angle and electromagnetic generator torque to control the wind turbine. In full load operation a feedback term from the collective blade pitch angle is used to schedule the gains of the proportional-integral controller to counter the effects of changing dynamics of the wind turbine for different wind speeds.
Blade pitch servo and generator models are not included in this controller and should be modeled separately, if they are to be included in the simulations.

Report Number: DTU Wind Energy E-0018
Publication Date: January 2013

Contract Number: EUDP project *Light Rotor*

Project Number: 46-43028-Xwp3

Sponsorship: -

ISBN: 978-87-92896-27-8

Cover: -

Pages: 42
Tables: 5
References: 3

Technical University of Denmark
DTU Wind Energy
Frederiksborgvej 399
4000 Roskilde
Denmark
Telephone +45 4677 5085

Contents

1. Controller description

This chapter contains a description of the controller strategy, architecture, filters, parameters, and in-/outputs. The controller is a further development of a previous controller used at DTU Wind Energy under the version 11. The new developments are inspired by Bossanyi's controller for the 5MW NREL reference turbine [1], where the power error feedback in the pitch controller that keeps the pitch at its minimum below rated power operation represents the most significant change.

The controller is only considering low speed shaft (LSS) measures of rotational speeds and torques, i.e., there is no gearbox on the drivetrain. The controller still be used for turbines with a gearbox, when the user transforms torques and speeds between the LSS and HSS using the gear ratio.

1.1. Strategy and architecture

A diagram of the entire controller is shown in Figure 1.1. The routes of this diagram that are active when the turbine is operating below rated power, herein called *partial load* operation, are shown in Figure 1.2. The routes that are active in *full load* operation are shown in Figure 1.3. These two regions of operation are first described before the switching between is explained. The discrete filters denoted by the functions f_1, f_2, f_p, and f_n in the diagrams are described and tested in Appendix B.

1.1.1. Partial load operation

The strategy for optimal C_P tracking in partial load operation is based on a balance between generator and aerodynamic torques to obtain a close to optimal tip speed ratio. To avoid the feedback of higher frequency dynamics (e.g. the drivetrain torsion mode), the torque reference $Q_{ref,k}$ at the current step k is computed based on a second order low-pass filtered LSS generator speed as $K\bar{\Omega}_k$. This feedback is enforced by setting the torque limits for the PID controller to $Q_{g,min,k} = Q_{g,max,k} = K\bar{\Omega}_k$ whenever the filtered rotational speed $\bar{\Omega}_k$ is not close to the minimum speed Ω_{min}, or the rated speed Ω_0. When the rotational speed is close to its bounds, these torque limits will open according to the interpolation factors $\sigma_{min,k}$ and $\sigma_{max,k}$. The torque reference will then be given by the PID controller based on the speed error $e_{Q,k} = \bar{\Omega}_k - \Omega_{set,k}$, where the set point is the minimum, or rated speed. Because the rotor speed is bounded, the power loss can often be minimized be performing some adjustment of the minimum pitch. A first order low-pass filtered wind speed measured at hub height $\bar{V}_k$ is used as parameter for varying the minimum pitch angle $\theta_{min,k} = \theta_{min}(\bar{V}_k)$ based on a look-up table provided by the user.

The power error feedback of pitch PID controller ensures that the pitch reference is kept at this minimum pitch angle.

The interpolation factors for the opening of the torque limits are based on how close the second order low-pass filtered generator speed is from its minimum and rated speeds. The limits can be opened gradually over an interval as described by the function

$$\sigma\left(x_0, x_1; x\right) = \begin{cases} 0 & \forall x < x_0 \\ a_3 x^3 + a_2 x^2 + a_1 x + a_0 & \forall x \in [x_0 : x_1[\\ 1 & \text{otherwise} \end{cases} \tag{1.1}$$

where the coefficients of the spline are

$$a_3 = \frac{2}{\left(x_0 - x_1\right)^3}, \quad a_2 = \frac{-3\left(x_0 + x_1\right)}{\left(x_0 - x_1\right)^3}, \quad a_1 = \frac{6 x_1 x_0}{\left(x_0 - x_1\right)^3}, \quad a_0 = \frac{\left(x_0 - 3x_1\right) x_0^2}{\left(x_0 - x_1\right)^3} \tag{1.2}$$

The function is programmed such that if $x_0 \geq x_1$ then the σ-function becomes

$$\sigma\left(x_0, x_0; x\right) = \begin{cases} 0 & \forall x < x_0 \\ 1 & \text{otherwise} \end{cases} \tag{1.3}$$

Figure 1.4 shows an example of the σ-function where $x_0 = 1$ and $x_1 = 2$. In an actual implementation of the controller, this smooth function with a third order polynomial may be an unnecessary complexity, which can replaced by a linear interpolation function.

Figure 1.5 shows the torque limits in partial load operation of the DTU 10 MW RWT, where the minimum speed is 7 rpm and rated speed is 9.6 rpm. The limits are set to be closed approximately 5 % above the minimum speed and start opening again at 90 % and are fully open at 95 % of the rated speed.

1.1.2. Full load operation

In full load operation, the torque limits are closed around the torque given by the selected power control strategy, either constant power P_0/Ω_k, or constant torque P_0/Ω_0, where P_0 is the rated power. Note that the unfiltered measured LSS generator speed Ω_k is used for computation of the reference torque in the constant power control.

The pitch reference angle is obtained from a combined PI feedback of the generator speed and power errors, and a possible differential feedback of the speed error. The speed error is obtained as the difference between the second order low-pass filtered LSS generator speed and the rated speed. The power error is the difference between the reference power $P_{ref,k} = Q_{ref,k}\Omega_k$ and the rated power P_0. Both errors are notch filtered around the frequency specified by the user as the free-free drivetrain frequency. This frequency is assumed to be constant although HAWCStab2 eigenvalue analysis often show a small variation with operational point (wind speed). Note that both errors contribute to the same proportional term ($\theta_{P,k}$) and same integral term ($\theta_{I,k}$). The latter is important because it ensures that the reference pitch angle is kept at the minimum pitch angle until rated power is reached; assuming that the right weighting between the integral speed error gain k_I and power error gain k_I^P has been selected by the user.

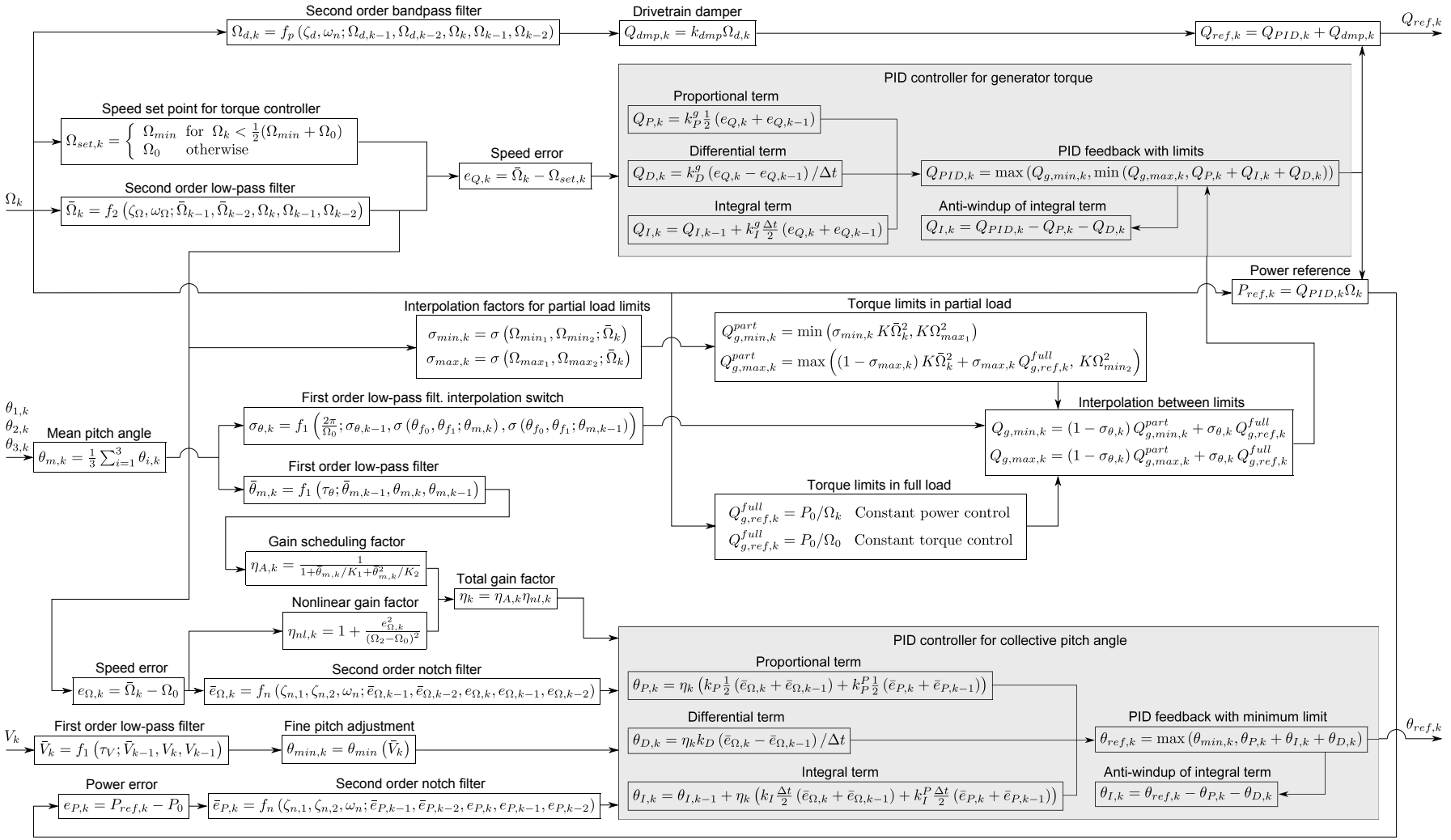

Figure 1.1.: Diagram of the discrete controller. Note that k denotes the current time step.

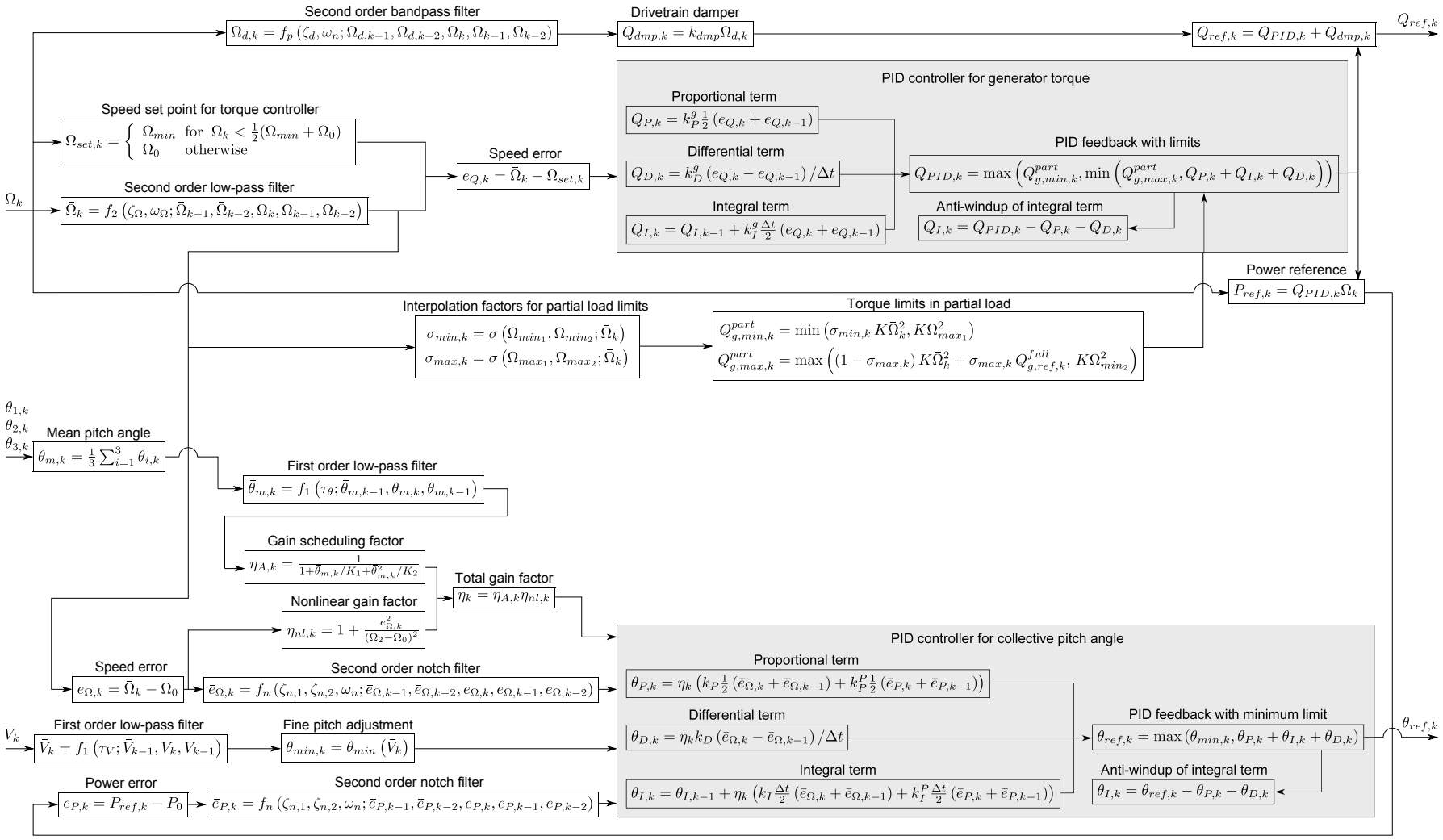

Figure 1.2.: Active routes during partial load operation in the controller diagram in Figure 1.1.

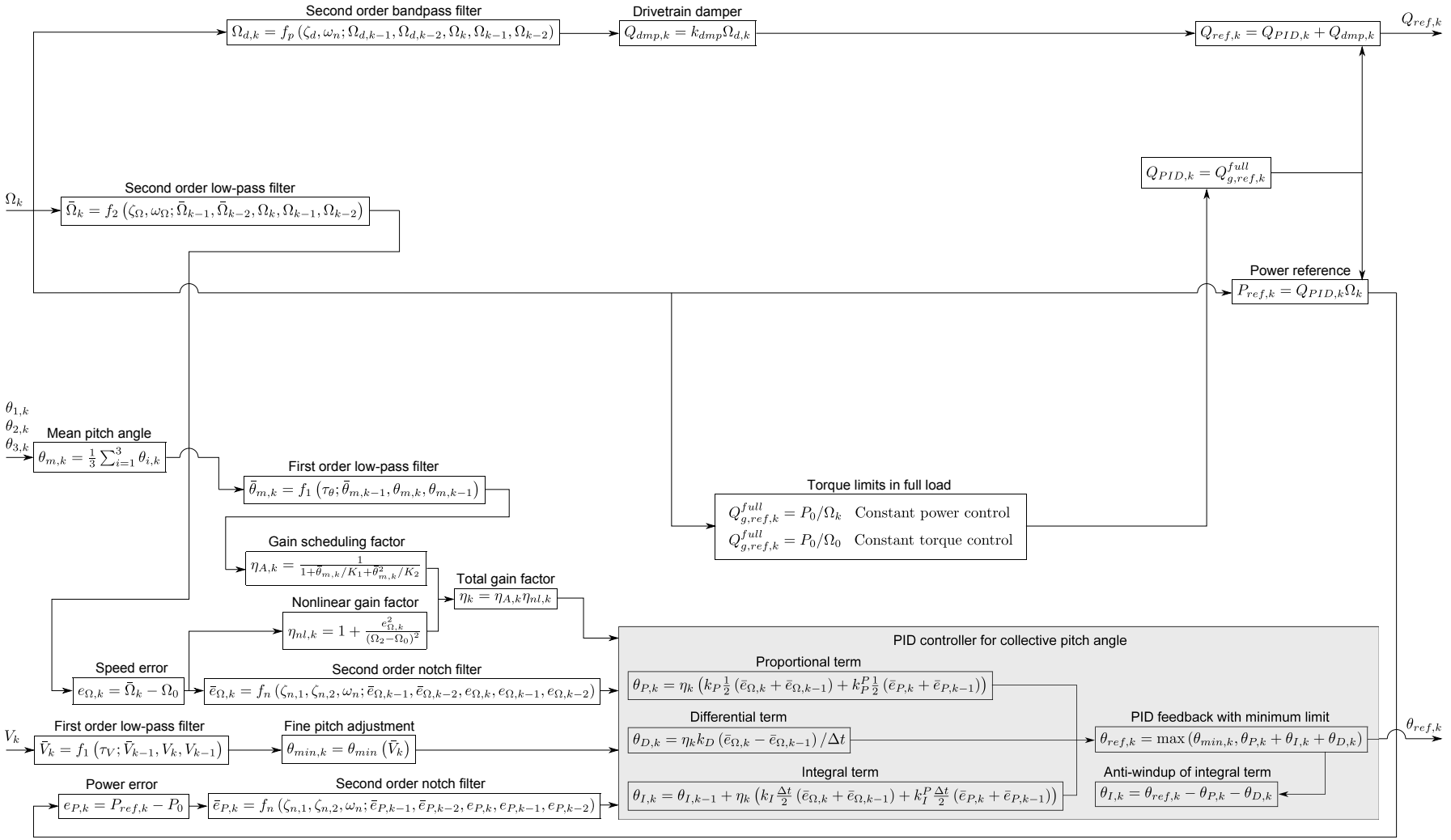

Figure 1.3.: Active routes during full load operation in the controller diagram in Figure 1.1.

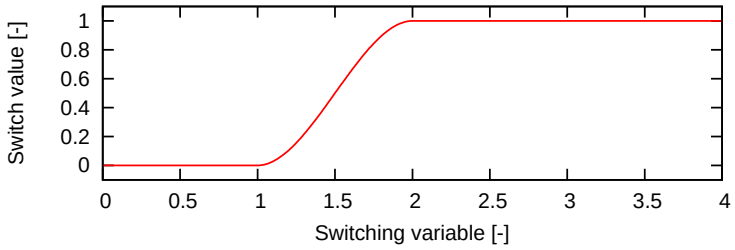

Figure 1.4.: Example of the σ-function (1.1) where $x_0 = 1$ and $x_1 = 2$.

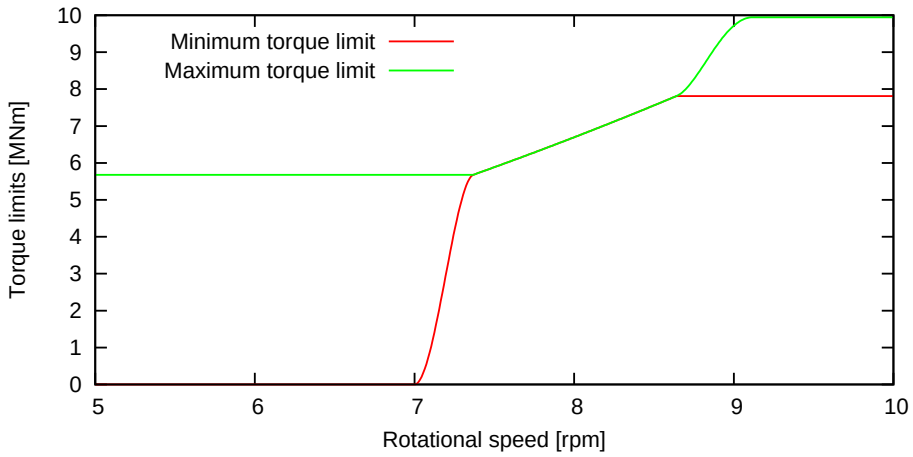

Figure 1.5.: Torque limits in partial load operation of the DTU 10 MW RWT, where the minimum speed is 7 rpm and rated speed is 9.6 rpm. The limits are set to be closed approximately 5 % above the minimum speed and start opening again at 90 % and are fully open at 95 % of the rated speed.

The anti-windup is performed such that the controller will react quickly with increased pitch angle if the reference power signal suddenly increased above the rated power level: In each time step, the minimum pitch limit is enforced on the reference pitch which is the sum of the proportional, differential, and integral terms $\theta_{ref,k} = \max(\theta_{min,k}, \theta_{P,k} + \theta_{D,k} + \theta_{I,k})$. The value of the integral term to be used for the integration in the next time step is then recalculated as $\theta_{I,k} = \theta_{ref,k} - \theta_{P,k} - \theta_{D,k}$, which only makes a change to the integral term if the reference pitch is on the minimum limit $\theta_{ref,k} = \theta_{min,k}$. Below rated power, where the proportional term is negative because the rotational speed error is kept close to zero by the torque PID controller, the integral term will therefore be positive. If the reference power is increased and becomes close to rated power (due to the reaction of the torque PID controller to an increased wind speed), then the proportional term will then come close to zero whereas the integral will still be positive and the resulting pitch reference angle will be positive, whereby large power and speed variations are avoided. Note that the same anti-windup scheme is used in the torque PID controller.

The first order low-pass filtered mean of the blade pitch angles $\bar{\theta}_{m,k}$ is used for scheduling of the gains of the pitch PID controller. A quadratic dependency of the aerodynamic

torque gain with collective pitch angle is assumed as

$$\frac{\partial Q_A}{\partial \theta} = \left.\frac{\partial Q_A}{\partial \theta}\right|_{\theta=0} \left(1 + \frac{\theta}{K_1} + \frac{\theta^2}{K_2}\right) \tag{1.4}$$

where Q_A denotes the aerodynamic torque, θ is the collective pitch angle, and $\left.\frac{\partial Q_A}{\partial \theta}\right|_{\theta=0}$ is the aerodynamic gain at zero pitch. The parameters of this expression K_1 and K_2 can be obtained from curve fitting to the derivative of the aerodynamic torque with respect to collective pitch angle assuming quasi-steady aerodynamics and *frozen wake* (constant induced velocities) as

$$\frac{\partial Q_A}{\partial \theta} = \frac{1}{2}\rho B \int_0^R c(r)U(r)^2 \left(C'_L(\alpha(r))\sin\varphi(r) - C'_D(\alpha(r))\cos\varphi(r)\right) r dr \tag{1.5}$$

where B is the number of blades, R is the outer radius of the rotor, $c(r)$ is the radial chord distribution, $U(r)$ is the mean steady state relative inflow velocity along the blade, C'_L and C'_D are the gradients of the lift and drag coefficient curves evaluated at the mean steady state angle of attack $\alpha(r)$ along the blade, and $\varphi(r)$ is the spanwise distribution of inflow angles relative to the rotor plane.

Figure 1.6 shows the aerodynamic torque gradients obtained from HAWCStab2 for the DTU 10 MW RWT together with the fit of the quadratic expression (1.4). The fitted are parameters shown as inputs number 21 and 22 in Table 1.1 of Section 1.2. Often, a linear fit is sufficient and it is assumed when the user enters $K_2 = 0$ (note that K_1 then is the angle where the aerodynamic gain is doubled). The gain scheduling factor based on the filtered mean pitch angle $\eta_{A,k}$ is the inverse of the expression in the parenthesis of (1.4). A nonlinear gain factor $\eta_{nl,k}$ based on the generator speed error is also added for increased sensitivity of the pitch PID controller by large speed excursions.

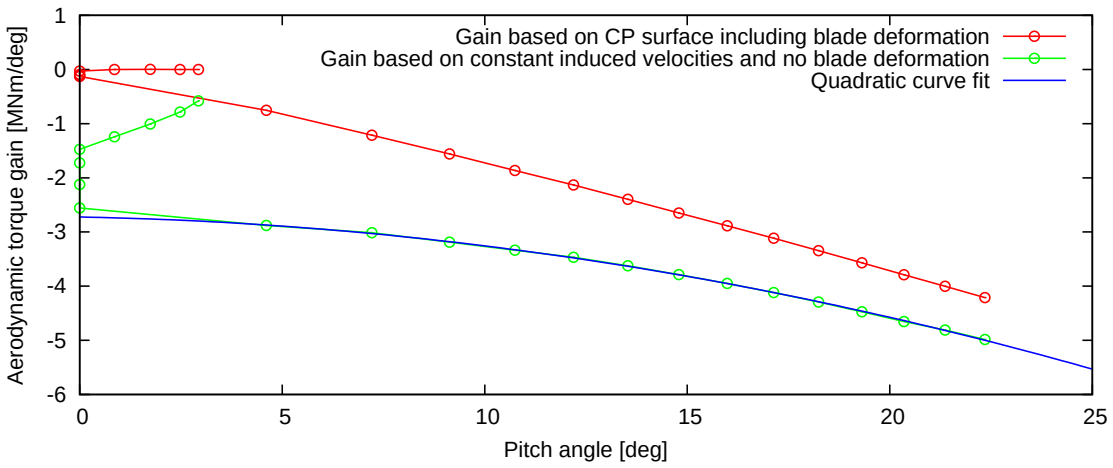

Figure 1.6.: Aerodynamic torque gains for the DTU 10 MW RWT obtained from HAWCStab2.

1.1.3. Switching between partial and full load operation

The switching between partial and full load control of the generator torque is based on a first order low-pass filtered switching variable $\sigma_{\theta,k}$ that is driven by a σ-function evaluation using the measured mean pitch angle θ_m. The time constant is the rotational period at rated speed. As explained above, the anti-windup of the combined integral term of the pitch PID controller will ensure that the reference pitch angle raises above its minimum value when the torque PID controller of generator speed results in a reference power close to the rated power level. The user can define at how many degrees above the minimum pitch this switching shall occur. Good experiences have been obtained with a hard switch at $\theta_{f_1} = \theta_{f_2} = \theta_{min,k} + 0.5$ deg.

1.1.4. Drivetrain damper

The measured generator LSS speed Ω_k is fed through a band-pass filter with the free-free drivetrain torsional frequency ω_n as center frequency (see example of the filter in Appendix B). The filtered speed $\Omega_{d,k}$, containing in-phase variations with frequencies around this frequency, is then multiplied by a gain factor k_{dmp} and added to the torque feedback from the PID controller to give the generator torque reference. Note that this drivetrain damper is always active when $k_{dmp} > 0$.

1.2. Parameters

All parameters of the controller are transferred to the DLL using HAWC2 commands for the "init" routine of the type2 DLL [2]. Table 1.1 shows the command list from the input to the DTU 10MW RWT, and some more details of all 38 parameters are given in Table 1.2, where the notation of the parameters used in the diagram of Figure 1.1 and some additional notations can also be seen.

1.3. Inputs and outputs

As input during simulations, the controller requires the time [s], the generator LSS speed [rad/s], the pitch angles of three blades [rad], and the three components of the wind speed at hub height [m/s], where the first and second of these three components must be the two horizontal components that are used internally to compute the horizontal vector sum. Table 1.4 shows the HAWC2 commands that provide these eight controller inputs for the DTU 10 MW RWT. Note that if the rotor only has two blades, then the controller may still be used by repeating a blade pitch angle output to the controller. However, this trick is only valid if there is no additional cycle, or individual pitch controller appended to this controller.

Table 1.5 contains a list of the controller outputs to the simulation, where the generator torque reference and pitch angle references for three blades are the

```
; Overall parameters
constant   1 10000.0  ; Rated power [kW]
constant   2   0.733  ; Minimum rotor speed [rad/s]
constant   3   1.005  ; Rated rotor speed [rad/s]
constant   4  15.6e6  ; Maximum allowable generator torque [Nm]
constant   5 100.0    ; Minimum pitch angle, theta_min [deg],
                      ; if |theta_min|>90, then a table of <wsp,theta_min> is read ;
                      ; from a file named 'wptable.n', where n=int(theta_min)
constant   6  90.0    ; Maximum pitch angle [deg]
constant   7  10.0    ; Maximum pitch velocity operation [deg/s]
constant   8   0.2    ; Frequency of generator speed filter [Hz]
constant   9   0.7    ; Damping ratio of speed filter [-]
constant  10   0.64   ; Frequency of free-free DT torsion mode [Hz], if zero no notch filter used
; Partial load control parameters
constant  11   9.5e6  ; Optimal Cp tracking K factor [kNm/(rad/s)^2], ;
                      ; Qg=K*Omega^2, K=eta*0.5*rho*A*Cp_opt*R^3/lambda_opt^3
constant  12   7.33e7 ; Proportional gain of torque controller [Nm/(rad/s)]
constant  13   1.32e7 ; Integral gain of torque controller [Nm/rad]
constant  14   0.0    ; Differential gain of torque controller [Nm/(rad/s^2)]
; Full load control parameters
constant  15   1      ; Generator control switch [1=constant power, 2=constant torque]
constant  16   0.592  ; Proportional gain of pitch controller [rad/(rad/s)]
constant  17   0.133  ; Integral gain of pitch controller [rad/rad]
constant  18   0.0    ; Differential gain of pitch controller [rad/(rad/s^2)]
constant  19   0.4e-8 ; Proportional power error gain [rad/W]
constant  20   0.4e-8 ; Integral power error gain [rad/(Ws)]
constant  21 164.13   ; Coefficient of linear term in aerodynamic gain scheduling, KK1 [deg]
constant  22 702.09   ; Coefficient of quadratic term in aerodynamic gain scheduling, KK2 [deg^2] &
                      ; (if zero, KK1 = pitch angle at double gain)
constant  23   1.3    ; Relative speed for double nonlinear gain [-]
; Cut-in simulation parameters
constant  24   0.1    ; Cut-in time [s]
constant  25   4.0    ; Time delay for soft start of torque [1/1P]
; Cut-out simulation parameters
constant  26   710    ; Cut-out time [s]
constant  27   5.0    ; Time constant for 1st order filter lag of torque cut-out [s]
constant  28   1      ; Stop type [1=linear two pitch speed stop, 2=exponential pitch speed stop]
constant  29   1.0    ; Time delay for pitch stop 1 [s]
constant  30  20.0    ; Maximum pitch velocity during stop 1 [deg/s]
constant  31   1.0    ; Time delay for pitch stop 2 [s]
constant  32  10.0    ; Maximum pitch velocity during stop 2 [deg/s]
; Expert parameters (keep default values unless otherwise given)
constant  33   0.5    ; Lower angle above lowest minimum pitch angle for switch [deg]
constant  34   0.5    ; Upper angle above lowest minimum pitch angle for switch [deg], if equal then hard switch
constant  35  95.0    ; Ratio between filtered speed and reference speed for fully open torque limits [%]
constant  36   5.0    ; Time constant of 1st order filter on wind speed used for minimum pitch [1/1P]
constant  37   5.0    ; Time constant of 1st order filter on pitch angle used for gain scheduling [1/1P]
; Drivetrain damper
constant  38   0.0  ; Proportional gain of active DT damper [Nm/(rad/s)], requires frequency in input 10
```

Table 1.1.: All parameters of the controller, here shown as the HAWC2 input commands for the "init" routine of the controller, see type2 DLL interface description in the HAWC2 manual. The shown values are taken from the input to the DTU 10MW RWT.

1.4. Cut-in procedure - start up at any wind speed

The controller has a very simplified cut-in procedure, which is not intended to model a real cut-in, but rather to enable start-up of normal operation DLCs at any wind speed. The blades are initially pitched out to maximum pitch and then at a given time in the

Input		Additional explanation if assumed needed
1	P_0	Rated power [kW].
2	Ω_{min}	Minimum rotor speed [rad/s].
3	Ω_0	Rated rotor speed [rad/s].
4	-	Maximum allowable generator torque [Nm]. An upper limit set on the torque reference signal.
5	-	This number is the minimum pitch angle θ_{min} in degrees, which is set to a constant if this input is less than 90 deg. Otherwise, the init routine will search for a file with the name "wptable.n", where "n" is a character string obtained from the integer value of the input. In the shown example, this file is therefore "wptable.100". The file format is first line contains an integer with the number of subsequent lines, which contain two numbers each, wind speed and minimum pitch angle in degrees. An example is shown in Table 1.3.
6	θ_{max}	Maximum pitch angle [deg].
7	-	Maximum pitch velocity operation [deg/s]. An upper limit set on the rate of change of the pitch reference signal.
8	ω_Ω	Frequency of generator speed filter [Hz].
9	ζ_Ω	Damping ratio of speed filter [-].
10	ω_n	Frequency of free-free DT torsion mode [Hz], if zero no notch filter used.
11	K	Optimal C_P tracking factor [Nm/(rad/s)^2], $K = \eta\frac{1}{2}\rho A C_{P,opt} R^3/\lambda_{opt}^3$.
12	k_P^g	Proportional gain of torque controller [Nm/(rad/s)].
13	k_I^g	Integral gain of torque controller [Nm/rad].
14	k_D^g	Differential gain of torque controller [Nm/(rad/s^2)].
15	-	Generator control strategy [1=constant power, 2=constant torque].
16	k_P	Proportional gain of pitch controller [rad/(rad/s)].
17	k_I	Integral gain of pitch controller [rad/rad].
18	k_D	Differential gain of pitch controller [rad/(rad/s^2)].
19	k_P^P	Proportional power error gain [rad/W].
20	k_I^P	Integral power error gain [rad/(Ws)].
21	K_1	Coefficient of linear term in aerodynamic gain scheduling [deg].
22	K_2	Coefficient of quadratic term in aerodynamic gain scheduling [deg^2]. If this factor K_2 is set to zero then the controller will assumed a linear gain scheduling. The K_1 is then the pitch angle where the aerodynamic torque gain has doubled from its value at zero pitch.
23	Ω_2/Ω_0	Normalized speed where the pitch controller gains are doubled.
24	-	Cut-in time [s], if zero no cut-in simulated.
25	-	A time delay for the cut-in procedure given in the unit [1/1P] corresponding to the rotational period at rated speed.
26	τ_{out}	Cut-out time [s], if zero no cut-out simulated.
27	-	Time constant for first order filter lag of torque cut-out [s].
28	-	Stop type [1=linear two pitch speed stop, 2=exponential pitch speed stop] as described in Section 1.5.
29	τ_1	Time delay for pitch stop 1 [s].
30	-	Maximum pitch velocity during stop 1 [deg/s].
31	τ_2	Time delay for pitch stop 2 [s].
32	-	Maximum pitch velocity during stop 2 [deg/s].

Input		Additional explanation if assumed needed
33	θ_{f_0}	Lower angle above lowest minimum pitch angle for switch [deg].
34	θ_{f_1}	Upper angle above lowest minimum pitch angle for switch [deg].
35	γ	Percentage of the rated speed when the torque limits are fully opened $\Omega_{max_2} = \gamma\Omega_0$ to let PID controller be active, and the opening starts at $\Omega_{min_2} = (2\gamma - 1)\Omega_0$. The same percentage is used for opening the torque limits for PID control around the minimum rotational speed, where the torque limits start to open at $\Omega_{max_1} = \Omega_{min}/\gamma$ and fully open at $\Omega_{min_1} = \Omega_{min}$.
36	$\tau_V\Omega_0/(2\pi)$	Time constant of 1st order filter on wind speed used for minimum pitch [1/1P].
37	$\tau_\theta\Omega_0/(2\pi)$	Time constant of 1st order filter on pitch angle for gain scheduling [1/1P].
38	k_{dmp}	Proportional gain of DT damper [Nm/(rad/s)], requires frequency in input 10.

Table 1.2.: All parameters of the controller related to the parameters shown in the diagram in Figure 1.1 and with additional explanations compared to Table 1.1.

```
7
 0.0 3.0
 4.0 3.0
 5.0 2.5
 6.0 1.7
 7.0 0.8
 8.0 0.0
50.0 0.0
```

Table 1.3.: Example of a "wptable.n" file. First line contains an integer with the number of subsequent lines, which contain two numbers each, wind speed and minimum pitch angle in degrees.

simulation (input 24), the blades are pitched towards minimum pitch with zero generator torque reference. As the rotor speed increases, a first order filter of the difference the measured rotational speed and the minimum rotational speed with the time constant of 1/1P (one rotational period at rated speed) is updated in each time step k as

$$\Delta\Omega_k = f_1\left(2\pi/\Omega_0; \Delta\Omega_{k-1}, \Omega_k - \Omega_{min}, \Omega_{k-1} - \Omega_{min}\right) \tag{1.6}$$

During this speed-up, the speed error PID terms of the pitch controller is active at a quarter of their normal gains and the set point of it is the minimum rotor speed. The power error gains are set to zero, otherwise it will stay at minimum due to the zero generator torque reference (the factor of a quarter is chosen based on trial and error). The pitch controller thereby catches the rotational speed at the minimum speed and when $\Delta\Omega_k$ become within 2 % of the minimum speed, the acceleration of the rotor is assumed to be under control and a generator cut-in time is registered. The torque reference is thereafter

```
general time                              ; [s]
constraint bearing1 shaft_rot 1 only 2 ; [rad/s] Generator LSS speed
constraint bearing2 pitch1 1 only 1    ; [rad]
constraint bearing2 pitch2 1 only 1    ; [rad]
constraint bearing2 pitch3 1 only 1    ; [rad]
wind free_wind 1 0.0 0.0 -124.6        ; [m/s] global coords at hub height
```

Table 1.4.: HAWC2 commands that define the input to the controller DLL. Note that the command "wind free_wind 1 x y z" will give all three components of the free wind at the point x,y,z, both in global coordinates [2], thus in all eight inputs.

ramped up using the σ-function in Equation (1.1) to its value determined by the normal controller at a user-defined time after the generator cut-in time. In the same period, the rotational speed set point for the pitch controller is ramped up to the rated speed Ω_0, and both speed and power error gains are ramped up to their values determined by the normal gain scheduling. The controller should thereafter be operating normally, and this start-up will at moderate to high wind speeds take less than 100 s for the DTU 10 MW RWT.

There should be no need for wind ramping for the controller to start up, however, caution should be made on not to start high wind speed (above rated) simulations with too high initial rotational speed. Too high initial rotor speed may cause the pitch and torque controllers to overreact and they may enter a state of competition. If the user can allow long start-up periods then the most stable way is set a late cut-in time to let the rotor slow down to idling before the cut-in starts. However, at low wind speeds, the start-up will then take a long time, and a very early cut-in (e.g. 0.1 s) is recommended, combined with an initial rotational speed is set to a value 50-75 % of the minimum speed.

1.5. Cut-out procedures

The user may specify a time in the simulation for the cut-out of the generator in input 26 (let it be denoted τ_{out}). A first order filter will be driven by the torque reference signal until this cut-out time, whereafter the filter is let to decay with a user specified time constant (input 27), and the torque reference will set equal to this decaying signal.

At the generator cut-out time plus the first user specified time delay in input 29 (let it be denoted τ_1), the blades will start to pitch out by setting the reference pitch angle to the maximum pitch angle (let it be denoted θ_{max}). The user can specify two different types of pitch velocity schemes, either linear pitching with two different constant speeds, or exponential pitching where the velocity changes as an exponential function. In the linear pitching scheme, the maximum pitch velocity is initial set to the pitch velocity specified in input 30. After additional time given by the "time delay of pitch stop 2" in input 31, let it be denoted τ_2, the maximum pitch velocity is then set to the pitch velocity specified in input 32. In the exponential pitch scheme, the maximum pitch velocity will start at the velocity given by θ_{max}/τ_2, whereafter this velocity will decay as the exponential function $\exp\left(-(t-\tau_{out}-\tau_1)/\tau_2\right)$ until $(t-\tau_{out}-\tau_1)/\tau_2 > 10$, or this decaying maximum pitch

Channel	Description	
1	Generator torque reference, $Q_{ref,k}$	[Nm]
2	Pitch angle reference of blade 1, $\theta_{ref,k}$	[rad]
3	Pitch angle reference of blade 2, $\theta_{ref,k}$	[rad]
4	Pitch angle reference of blade 3, $\theta_{ref,k}$	[rad]
5	Power reference, $P_{ref,k}$	[W]
6	Filtered wind speed, $\bar{V}_k$	[m/s]
7	Filtered rotor speed, $\bar{\Omega}_k$	[rad/s]
8	Filtered rotor speed error for torque, $e_{Q,k}$	[rad/s]
9	Bandpass filtered rotor speed, $\Omega_{d,k}$	[rad/s]
10	Proportional term of torque controller, $Q_{P,k}$	[Nm]
11	Integral term of torque controller, $Q_{I,k}$	[Nm]
12	Minimum limit of torque, $Q_{g,min,k}$	[Nm]
13	Maximum limit of torque, $Q_{g,max,k}$	[Nm]
14	Torque limit switch based on pitch, $\sigma_{\theta,k}$	[-]
15	Low-pass filtered rotor speed error for pitch controller, $e_{\Omega,k}$	[rad/s]
16	Low-pass and notch filtered power error for pitch controller, $\bar{e}_{P,k}$	[W]
17	Proportional term of pitch controller, $\theta_{P,k}$	[rad]
18	Integral term of pitch controller, $\theta_{I,k}$	[rad]
19	Minimum limit of pitch, $\theta_{min,k}$	[rad]
20	Maximum limit of pitch, θ_{max}	[rad]
21	Torque reference from DT damper, $Q_{dmp,k}$	[Nm]

Table 1.5.: Outputs from the controller DLL, where only the first two are needed. The rest are for analysis of controller behavior.

velocity is lower than the pitch velocity specified in input 32, which is the value that the maximum pitch velocity will have thereafter.

1.6. Programming

The controller is programmed in Fortran90 using the format of the *type2 DLL interface for HAWC2* [2] and the source code is listed in Appendix A.

2. Simulations of DTU 10 MW RWT

Figure 2.1 – 2.6 shows selected signals from IEC NTM simulations with HAWC2 of the class 1A DTU 10 MW RWT at 4, 8, 12, 16, 20, and 24 m/s. All simulations are only meant as an example of the controller performance because they are performed using an earlier version of the DTU 10 MW RWT than the published one. The plots include the transients, which for all but lowest wind speed of 4 m/s have died out before 100 s into the simulations. Comments to simulations are given in the captions.

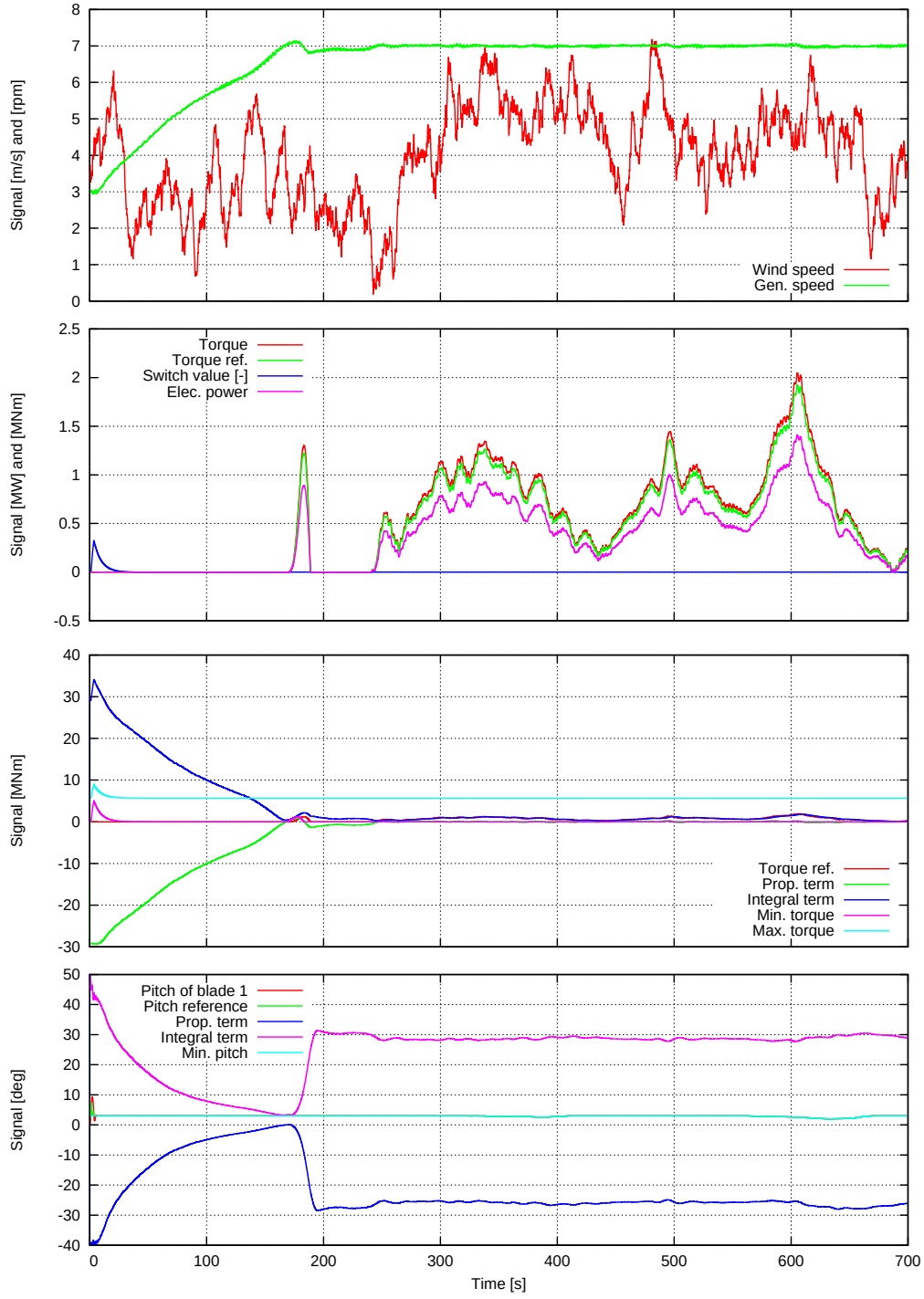

Figure 2.1.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 4 m/s. Note the long start-up time (transients) which for low wind speeds can be reduced by setting a higher initial rotor speed. Note also that a minimum pitch angle varies and that the integral pitch term is positive.

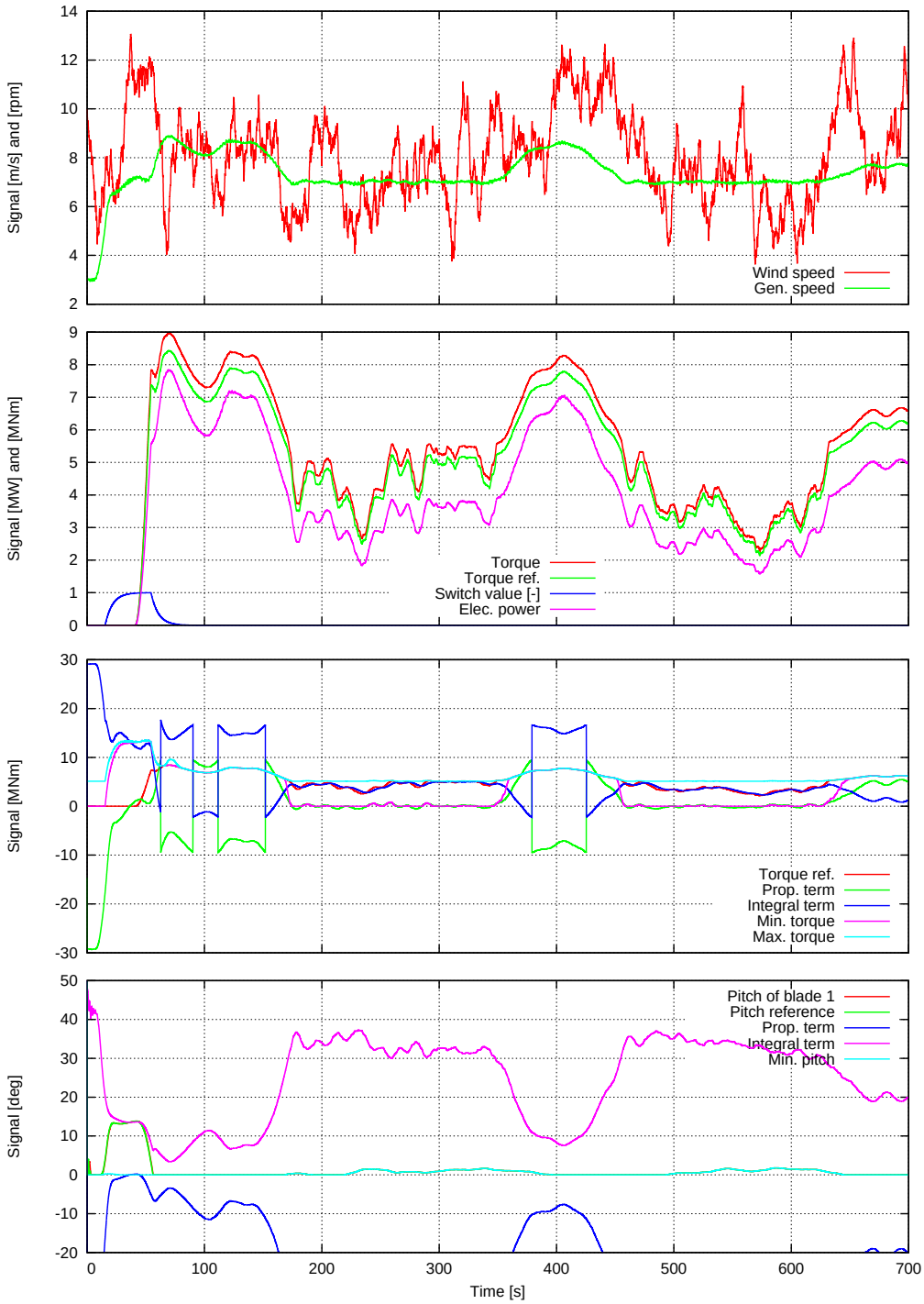

Figure 2.2.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 8 m/s. Note the generator is mainly running at its minimum speed. As the speed increases, the speed set point switches at the speed crosses the averaged between rated and this minimum speed, which causes the discontinuities on the torque limits. The variation of minimum pitch angle with wind speed is also seen. The integral pitch term is still positive.

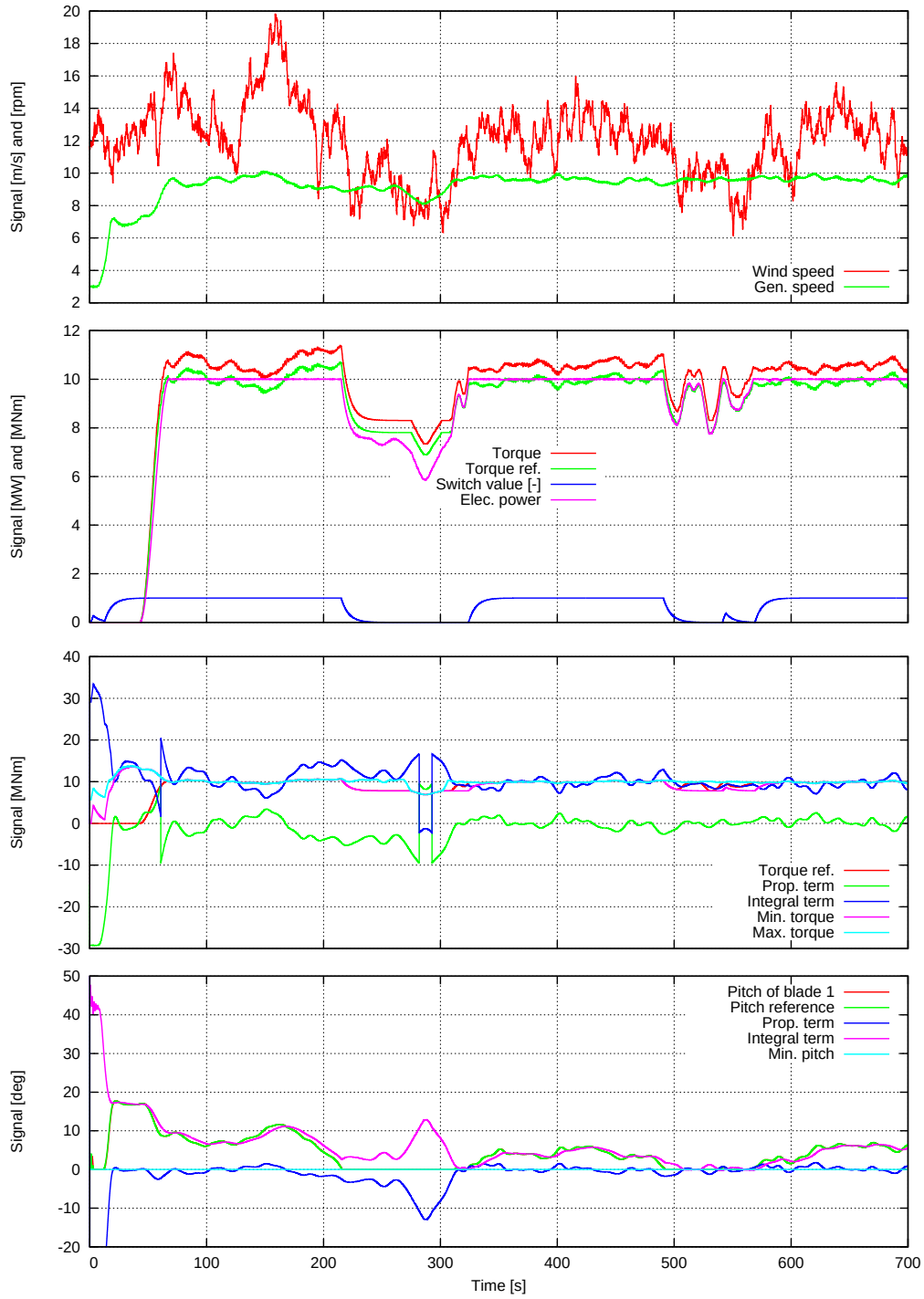

Figure 2.3.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 12 m/s. At this close to rated wind speeds, the turbine is starting to produce full power. During the transients, the pitch controller controls the rotor speed and the larger-than-minimum pitch results in a switch value of 1, whereby the torque reference is set to constant power control. Around 200 s, the wind speed has lowered, resulting in a lowering rotational speed, the pitch controller lowers the pitch angle, and eventually the switch goes to zero, i.e., partial load operation.

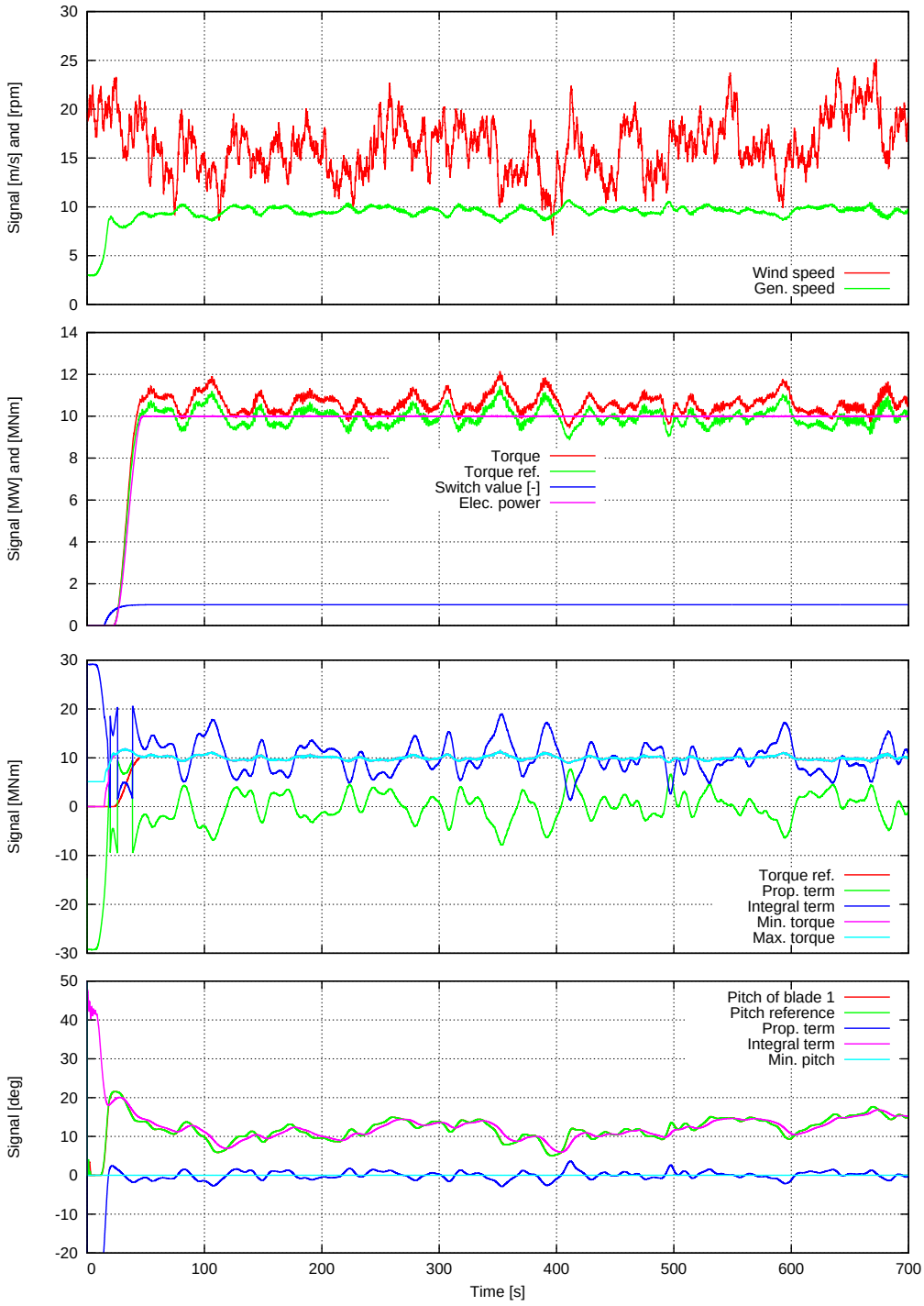

Figure 2.4.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 16 m/s. The turbine stays in full load operation.

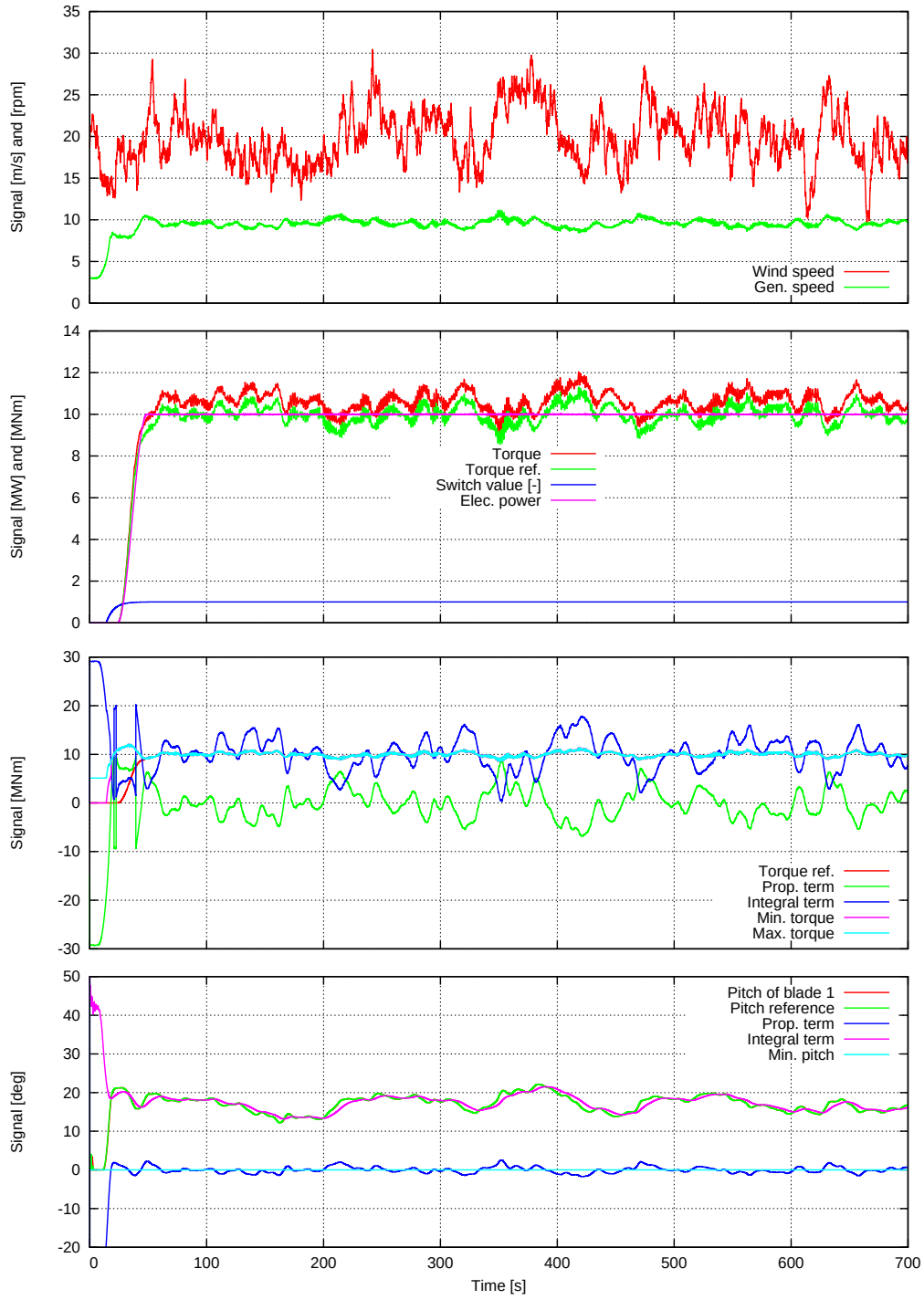

Figure 2.5.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 20 m/s. The turbine stays in full load operation.

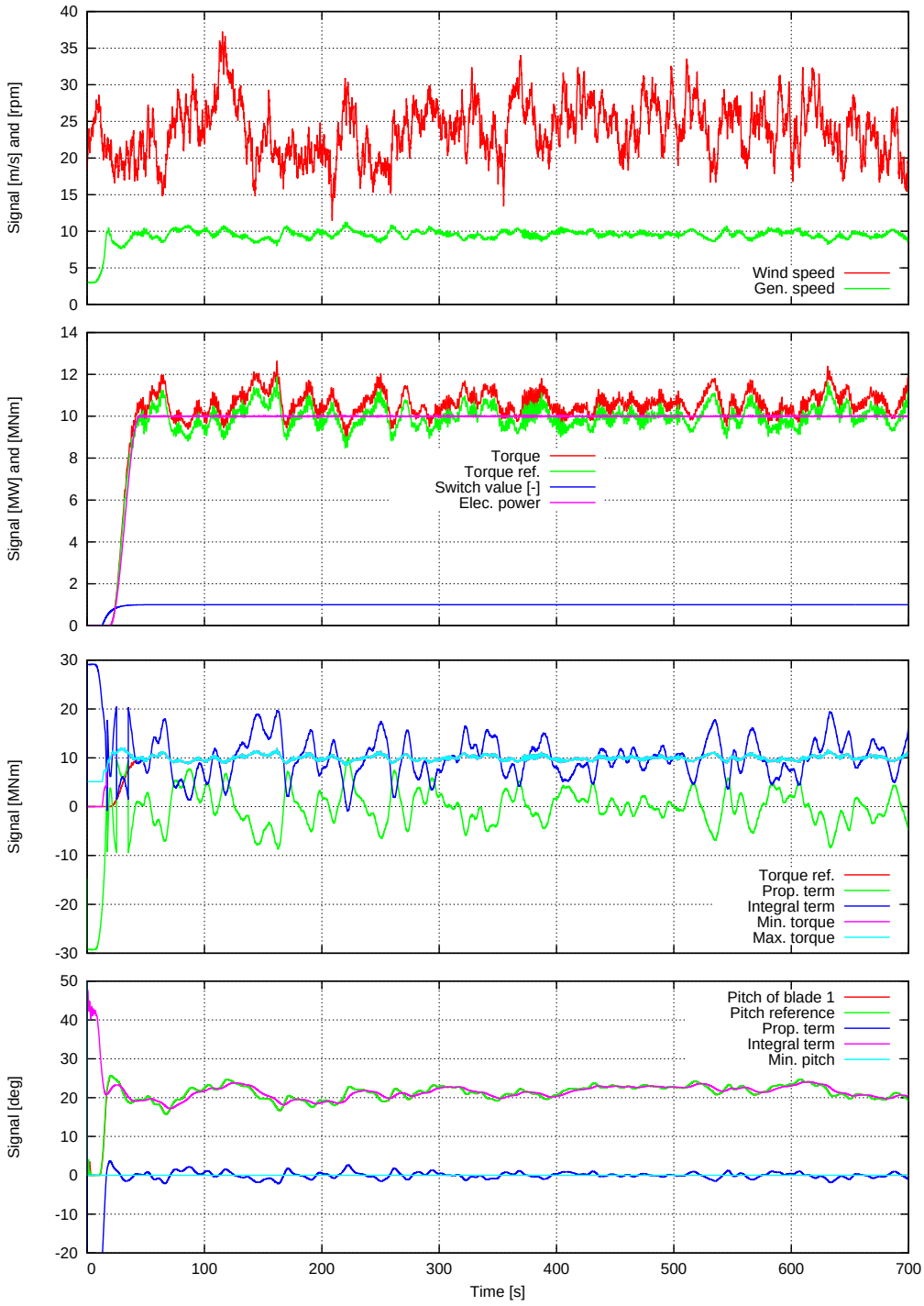

Figure 2.6.: Selected signals from IEC NTM simulation of the class 1A DTU 10 MW RWT at 24 m/s. The turbine stays in full load operation.

Bibliography

[1] E. Bossanyi. Controller for 5mw reference turbine. Technical Report 11593/BR/04, Garrad Hassan and Partners Limited, Bristol, England, July 2009. UpWind report.

[2] T. J. Larsen and A. M. Hansen. *How 2 HAWC2, the user's manual*. DTU Wind Energy, Roskilde, Denmark, 4.3 edition, April 2012. Risø-R-1597.

[3] A. H. Nayfeh and B. Balachandran. *Applied Nonlinear Dynamics*. Wiley, 1995.

Appendix A.

Source code

This appendix contains the source of the controller as given by revision no. 6 to the SVN repository

`//repos.gbar.dtu.dk/mhha/Controller/`

There are two files, risoe_controller.f90 and risoe_controller_fcns.f90 in this Fortran90 code. This type2 DLL for HAWC2 has been compiled with Intel Fortran 13.0.

A.1. Main routines in risoe_controller.f90

```
subroutine init_regulation(array1,array2)
use risoe_controller_fcns
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, C, ALIAS:'init_regulation'::init_regulation
real*8 array1(1000),array2(1)
! Local vars
integer*4 i,ifejl
character text32*32
real*8 minimum_pitch_angle
logical findes
! Input array1 must contain
!
!       ; Overall parameters
!    1: constant   1   ; Rated power [kW]
!    2: constant   2   ; Minimum rotor speed [rad/s]
!    3: constant   3   ; Rated rotor speed [rad/s]
!    4: constant   4   ; Maximum allowable generator torque [Nm]
!    5: constant   5   ; Minimum pitch angle, theta_min [deg],
!                      ; if |theta_min|>90, then a table of <wsp,theta_min> is read
!                      ; from a file named 'wptable.n', where n=int(theta_min)
!    6: constant   6   ; Maximum pitch angle [deg]
!    7: constant   7   ; Maximum pitch velocity operation [deg/s]
!    8: constant   8   ; Frequency of generator speed filter [Hz]
!    9: constant   9   ; Damping ratio of speed filter [-]
!   10: constant  10   ; Frequency of free-free DT torsion mode [Hz], if zero no notch filter used
!       ; Partial load control parameters
!   11: constant  11   ; Optimal Cp tracking K factor [Nm/(rad/s)^2],
!                      ; Qg=K*Omega^2, K=eta*0.5*rho*A*Cp_opt*R^3/lambda_opt^3
!   12: constant  12   ; Proportional gain of torque controller [Nm/(rad/s)]
!   13: constant  13   ; Integral gain of torque controller [Nm/rad]
!   14: constant  14   ; Differential gain of torque controller [Nm/(rad/s^2)]
!       ; Full load control parameters
!   15: constant  15   ; Generator control switch [1=constant power, 2=constant torque]
!   16: constant  16   ; Proportional gain of pitch controller [rad/(rad/s)]
!   17: constant  17   ; Integral gain of pitch controller [rad/rad]
!   18: constant  18   ; Differential gain of pitch controller [rad/(rad/s^2)]
```

```
!   19: constant  19   ; Proportional power error gain [rad/W]
!   20: constant  20   ; Integral power error gain [rad/(Ws)]
!   21: constant  21   ; Coefficient of linear term in aerodynamic gain scheduling, KK1 [deg]
!   22: constant  22   ; Coefficient of quadratic term in aerodynamic gain scheduling, KK2 [deg^2]
!                      ; (if zero, KK1 = pitch angle at double gain)
!   23: constant  23   ; Relative speed for double nonlinear gain [-]
!       ; Cut-in simulation parameters
!   24: constant  24   ; Cut-in time [s], if zero no cut-in simulated
!   25: constant  25   ; Time delay for soft start [1/1P]
!       ; Cut-out simulation parameters
!   26: constant  26   ; Cut-out time [s], if zero no cut-out simulated
!   27: constant  27   ; Time constant for 1st order filter lag of torque cut-out [s]
!   28: constant  28   ; Stop type [1=linear two pitch speed stop, 2=exponential pitch speed stop]
!   29: constant  29   ; Time delay for pitch stop 1 [s]
!   30: constant  30   ; Maximum pitch velocity during stop 1 [deg/s]
!   31: constant  31   ; Time delay for pitch stop 2 [s]
!   32: constant  32   ; Maximum pitch velocity during stop 2 [deg/s]
!       ; Expert parameters (keep default values unless otherwise given)
!   33  constant  33   ; Lower angle above lowest minimum pitch angle for switch [deg]
!   34: constant  34   ; Upper angle above lowest minimum pitch angle for switch [deg]
!   35: constant  35   ; Ratio between filtered and reference speed for fully open torque limits [%]
!   36: constant  36   ; Time constant of 1st order filter on wind speed used for minimum pitch [1/1P]
!   37: constant  37   ; Time constant of 1st order filter on pitch angle for gain scheduling [1/1P]
!   38: constant  38   ; Proportional gain of DT damper [Nm/(rad/s)], requires frequency in input 10
!
! Output array2 contains nothing for init
!
! Overall parameters
Pe_rated                  =array1( 1)*1.d3
omega_ref_min             =array1( 2)
omega_ref_max             =array1( 3)
max_lss_torque            =array1( 4)
minimum_pitch_angle       =array1( 5)*degrad
pitch_stopang             =array1( 6)*degrad
PID_pit_var.velmax        =array1( 7)*degrad
omega2ordervar.f0         =array1( 8)
omega2ordervar.zeta        =array1( 9)
DT_mode_filt.f0           =array1(10)
! Partial load control parameters
Kopt                      =array1(11)
if (Kopt*omega_ref_max**2.ge.Pe_rated/omega_ref_max) Kopt=Pe_rated/omega_ref_max**3
PID_gen_var.Kpro          =array1(12)
PID_gen_var.Kint          =array1(13)
PID_gen_var.Kdif          =array1(14)
! Full load control parameters
const_power          =(int(array1(15)).eq.1)
PID_pit_var.kpro(1)       =array1(16)
PID_pit_var.kint(1)       =array1(17)
PID_pit_var.kdif(1)       =array1(18)
PID_pit_var.kpro(2)       =array1(19)
PID_pit_var.kint(2)       =array1(20)
PID_pit_var.kdif(2)       =0.d0
kk1                       =array1(21)*degrad
kk2                       =array1(22)*degrad*degrad
rel_limit                 =array1(23)
! Cut-in simulation parameters
t_cutin                   =array1(24)
t_cutin_delay             =array1(25)*2.d0*pi/omega_ref_max
! Cut-out simulation parameters
t_cutout                  =array1(26)
torquefirstordervar.tau   =array1(27)
pitch_stoptype        =int(array1(28))
pitch_stopdelay           =array1(29)
pitch_stopvelmax          =array1(30)*degrad
pitch_stopdelay2          =array1(31)
pitch_stopvelmax2         =array1(32)*degrad
```

```
! Expert parameters (keep default values unless otherwise given)
switch1_pitang_lower     =array1(33)*degrad
switch1_pitang_upper     =array1(34)*degrad
rel_sp_open_Qg           =array1(35)*1.d-2
wspfirstordervar.tau     =array1(36)*2.d0*pi/omega_ref_max
pitchfirstordervar.tau   =array1(37)*2.d0*pi/omega_ref_max
! Drivetrain damper
DT_damp_gain             =array1(38)
DT_damper_filt.f0        =DT_mode_filt.f0
pwr_DT_mode_filt.f0      =DT_mode_filt.f0
! Default and derived parameters
PID_gen_var.velmax=0.d0 !No limit to generator torque change rate
Qg_rated=Pe_rated/omega_ref_max
switchfirstordervar.tau=2.d0*pi/omega_ref_max
cutinfirstordervar.tau=2.d0*pi/omega_ref_max
! Wind speed table
if (dabs(minimum_pitch_angle).lt.90.d0*degrad) then
  Opdatavar.lines=2
  Opdatavar.wpdata(1,1)=0.d0
  Opdatavar.wpdata(2,1)=99.d0
  Opdatavar.wpdata(1,2)=minimum_pitch_angle
  Opdatavar.wpdata(2,2)=minimum_pitch_angle
else
  write(text32,'(i)') int(minimum_pitch_angle*raddeg)
  inquire(file='wpdata.'//trim(adjustl(text32)),exist=findes)
  if (findes) then
    open(88,file='wpdata.'//trim(adjustl(text32)))
    read(88,*,iostat=ifejl) Opdatavar.lines
    if (ifejl.eq.0) then
      do i=1,Opdatavar.lines
        read(88,*,iostat=ifejl) Opdatavar.wpdata(i,1),Opdatavar.wpdata(i,2)
        if (ifejl.ne.0) then
          write(6,*) ' *** ERROR *** Could not read lines in minimum '&
                   //'pitch table in file wpdata.'//trim(adjustl(text32))
          stop
        endif
        Opdatavar.wpdata(i,2)=Opdatavar.wpdata(i,2)*degrad
      enddo
    else
      write(6,*) ' *** ERROR *** Could not read number of lines '&
                //'in minimum pitch table in file wpdata.'//trim(adjustl(text32))
      stop
    endif
    close(88)
  else
    write(6,*) ' *** ERROR *** File ''wpdata.'//trim(adjustl(text32))&
              //''' does not exist in the working directory'
    stop
  endif
endif
! Initiate the dynamic variables
stepno=0
time_old=0.d0
! No output
array2=0.d0
return
end subroutine init_regulation
!**************************************************************************************************
subroutine update_regulation(array1,array2)
use risoe_controller_fcns
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, C, ALIAS:'update_regulation'::update_regulation
real*8 array1(1000),array2(100)
! Input array1 must contain
!
!    1: general time                          ; [s]
```

```
!    2: constraint bearing1 shaft_rot 1 only 2 ; [rad/s] Generator LSS speed
!    3: constraint bearing2 pitch1 1 only 1    ; [rad]
!    4: constraint bearing2 pitch2 1 only 1    ; [rad]
!    5: constraint bearing2 pitch3 1 only 1    ; [rad]
!  6-8: wind free_wind 1 0.0 0.0 hub height    ; [m/s] global coords at hub height
!
! Output array2 contains
!
!    1: Generator torque reference            [Nm]
!    2: Pitch angle reference of blade 1      [rad]
!    3: Pitch angle reference of blade 2      [rad]
!    4: Pitch angle reference of blade 3      [rad]
!    5: Power reference                       [W]
!    6: Filtered wind speed                   [m/s]
!    7: Filtered rotor speed                  [rad/s]
!    8: Filtered rotor speed error for torque [rad/s]
!    9: Bandpass filtered rotor speed         [rad/s]
!   10: Proportional term of torque contr.    [Nm]
!   11: Integral term of torque controller    [Nm]
!   12: Minimum limit of torque               [Nm]
!   13: Maximum limit of torque               [Nm]
!   14: Torque limit switch based on pitch    [-]
!   15: Filtered rotor speed error for pitch  [rad/s]
!   16: Power error for pitch                 [W]
!   17: Proportional term of pitch controller [rad]
!   18: Integral term of pitch controller     [rad]
!   19: Minimum limit of pitch                [rad]
!   20: Maximum limit of pitch                [rad]
!   21: Torque reference from DT damper       [Nm]
!
! Local variables
real*8 time,omega,omegafilt,domega_dt_filt,wsp,WSPfilt
real*8 omega_err_filt_pitch,omega_err_filt_speed,omega_dtfilt
real*8 ommin1,ommin2,ommax1,ommax2
real*8 pitang(3),meanpitang,meanpitangfilt,theta_min,e_pitch(2)
real*8 kgain_pitch(3,2),kgain_torque(3),aero_gain,x,dummy,y(2)
real*8 Qg_min_partial,Qg_max_partial,Qg_min_full,Qg_max_full
real*8 Qgen_ref,theta_col_ref,thetaref(3),Pe_ref,Qdamp_ref
!************************************************************************************************
! Increment time step (may actually not be necessary in type2 DLLs)
!************************************************************************************************
time=array1(1)
if (time.gt.time_old) then
  deltat=time-time_old
  time_old=time
  stepno=stepno+1
endif
!************************************************************************************************
! Inputs and their filtering
!************************************************************************************************
omega=array1(2)
! Mean pitch angle
pitang(1)=array1(3)
pitang(2)=array1(4)
pitang(3)=array1(5)
meanpitang=(pitang(1)+pitang(2)+pitang(3))/3.d0
! Wind speed as horizontal vector sum
wsp=dsqrt(array1(6)**2+array1(7)**2)
! Low-pass filtering of the rotor speed
y=lowpass2orderfilt(deltat,stepno,omega2ordervar,omega)
omegafilt=y(1)
domega_dt_filt=y(2)
! Mean pitch angle
! Low-pass filtering of the mean pitch angle for gain scheduling
meanpitangfilt=min(lowpass1orderfilt(deltat,stepno,pitchfirstordervar,meanpitang),30.d0*degrad)
! Low-pass filtering of the nacelle wind speed
```

```
WSPfilt=lowpass1orderfilt(deltat,stepno,wspfirstordervar,wsp)
! Minimum pitch angle may vary with filtered wind speed
theta_min=GetOptiPitch(WSPfilt)
switch1_pitang_lower=switch1_pitang_lower+theta_min
switch1_pitang_upper=switch1_pitang_upper+theta_min
!************************************************************************************************
! Speed ref. changes max. <-> min. for torque controller and remains at rated for pitch controller
!************************************************************************************************
if (omegafilt.gt.0.5d0*(omega_ref_max+omega_ref_min)) then
  omega_err_filt_speed=omegafilt-omega_ref_max
else
  omega_err_filt_speed=omegafilt-omega_ref_min
endif
!************************************************************************************************
! PID regulation of generator torque
!************************************************************************************************
! Limits for full load
if (const_power) then
  Qg_min_full=dmin1(Pe_rated/dmax1(omega,1.d-15),max_lss_torque)
  Qg_max_full=Qg_min_full
else
  Qg_min_full=Qg_rated
  Qg_max_full=Qg_rated
endif
! Limits for partial load that opens in both ends
ommin1=omega_ref_min
ommin2=omega_ref_min/rel_sp_open_Qg
ommax1=(2.d0*rel_sp_open_Qg-1.d0)*omega_ref_max
ommax2=rel_sp_open_Qg*omega_ref_max
x=switch_spline(omegafilt,ommin1,ommin2)
Qg_min_partial=dmin1(Kopt*omegafilt**2*x,Kopt*ommax1**2)
x=switch_spline(omegafilt,ommax1,ommax2)
Qg_max_partial=dmax1(Kopt*omegafilt**2*(1.d0-x)+Qg_max_full*x,Kopt*ommin2**2)
! Switch based on pitch
switch1=switch_spline(meanpitang,switch1_pitang_lower,switch1_pitang_upper)
switch1=lowpass1orderfilt(deltat,stepno,switchfirstordervar,switch1)
! Interpolation between partial and full load torque limits based on switch 1
PID_gen_var.outmin=(1.d0-switch1)*Qg_min_partial+switch1*Qg_min_full
PID_gen_var.outmax=(1.d0-switch1)*Qg_max_partial+switch1*Qg_max_full
if (PID_gen_var.outmin.gt.PID_gen_var.outmax) PID_gen_var.outmin=PID_gen_var.outmax
! Compute PID feedback to generator torque
kgain_torque=1.d0
Qgen_ref=PID(stepno,deltat,kgain_torque,PID_gen_var,omega_err_filt_speed)
!------------------------------------------------------------------------------------------------
! Control of cut-in regarding generator torque
!------------------------------------------------------------------------------------------------
if (t_cutin.gt.0.d0) then
  if (generator_cutin) then
    x=switch_spline(time,t_generator_cutin,t_generator_cutin+t_cutin_delay)
    Qgen_ref=Qgen_ref*x
  else
    Qgen_ref=0.d0
  endif
endif
!------------------------------------------------------------------------------------------------
! Control of cut-out regarding generator torque
!------------------------------------------------------------------------------------------------
if ((t_cutout.gt.0.d0).and.(time.gt.t_cutout)) then
  Qgen_ref=lowpass1orderfilt(deltat,stepno,torquefirstordervar,0.d0)
else
  dummy=lowpass1orderfilt(deltat,stepno,torquefirstordervar,Qgen_ref)
endif
!------------------------------------------------------------------------------------------------
! Reference electrical power
!------------------------------------------------------------------------------------------------
Pe_ref=Qgen_ref*omega
```

```
!-------------------------------------------------------------------------------------------------
! Active DT damping based on notch filtered of rotor speed
!-------------------------------------------------------------------------------------------------
if ((DT_damp_gain.gt.0.d0).and.(DT_damper_filt.f0.gt.0.d0)) then
  omega_dtfilt=bandpassfilt(deltat,stepno,DT_damper_filt,omega)
  if (t_cutin.gt.0.d0) then
    if (generator_cutin) then
      x=switch_spline(time,t_generator_cutin+t_cutin_delay,t_generator_cutin+2.d0*t_cutin_delay)
      Qdamp_ref=DT_damp_gain*omega_dtfilt*x
      Qgen_ref=dmin1(dmax1(Qgen_ref+Qdamp_ref,0.d0),max_lss_torque)
    endif
  else
    Qdamp_ref=DT_damp_gain*omega_dtfilt
    Qgen_ref=dmin1(dmax1(Qgen_ref+Qdamp_ref,0.d0),max_lss_torque)
  endif
endif
!*************************************************************************************************
! PID regulation of collective pitch angle
!*************************************************************************************************
! Reference speed is equal rated speed
omega_err_filt_pitch=omegafilt-omega_ref_max
! Limits
PID_pit_var.outmin=theta_min
PID_pit_var.outmax=pitch_stopang
! Aerodynamic gain scheduling
if (kk2.gt.0.d0) then
  aero_gain=1.d0+meanpitangfilt/kk1+meanpitangfilt**2/kk2
else
  aero_gain=1.d0+meanpitangfilt/kk1
endif
! Nonlinear gain to avoid large rotor speed excursion
kgain_pitch=(omega_err_filt_pitch**2/(omega_ref_max*(rel_limit-1.d0))**2+1.d0)/aero_gain
!-------------------------------------------------------------------------------------------------
! Control of cut-in regarding pitch
!-------------------------------------------------------------------------------------------------
if (t_cutin.gt.0.d0) then
  if (time.lt.t_cutin) then
    PID_pit_var.outmin=pitch_stopang
    PID_pit_var.outmax=pitch_stopang
    kgain_pitch=0.d0
    dummy=lowpass1orderfilt(deltat,stepno,cutinfirstordervar,omega-omega_ref_min)
  else
    if (.not.generator_cutin) then
      kgain_pitch(1:3,1)=0.25d0*kgain_pitch(1:3,1)
      kgain_pitch(1:3,2)=0.d0
      omega_err_filt_pitch=omegafilt-omega_ref_min
      x=lowpass1orderfilt(deltat,stepno,cutinfirstordervar,omega-omega_ref_min)
      if (dabs(x).lt.omega_ref_min*1.d-2) then
        generator_cutin=.true.
        t_generator_cutin=time
      endif
    else
      x=switch_spline(time,t_generator_cutin,t_generator_cutin+t_cutin_delay)
      omega_err_filt_pitch=omegafilt-(omega_ref_min*(1.d0-x)+omega_ref_max*x)
      kgain_pitch(1:3,1)=0.25d0*kgain_pitch(1:3,1)+kgain_pitch(1:3,1)*0.75d0*x
      kgain_pitch(1:3,2)=kgain_pitch(1:3,2)*x
    endif
  endif
endif
!-------------------------------------------------------------------------------------------------
! Control of cut-out regarding pitch
!-------------------------------------------------------------------------------------------------
if ((t_cutout.gt.0.d0).and.(time.gt.t_cutout+pitch_stopdelay)) then
  select case(pitch_stoptype)
    case(1) ! Normal 2-step stop situation
      PID_pit_var.outmax=pitch_stopang
```

```
      PID_pit_var.outmin=pitch_stopang
      if (time.gt.t_cutout+pitch_stopdelay+pitch_stopdelay2) then
        PID_pit_var.velmax=pitch_stopvelmax2
      else
        PID_pit_var.velmax=pitch_stopvelmax
      endif
    case(2) ! Exponential decay approach
      PID_pit_var.outmax=pitch_stopang
      PID_pit_var.outmin=pitch_stopang
      if ((time-(t_cutout+pitch_stopdelay))/pitch_stopdelay2.lt.10.d0) then
        PID_pit_var.velmax=pitch_stopang/pitch_stopdelay2&
                          *dexp(-(time-(t_cutout+pitch_stopdelay))/pitch_stopdelay2)
      else
        PID_pit_var.velmax=0.d0
      endif
      if (PID_pit_var.velmax.gt.pitch_stopvelmax) PID_pit_var.velmax=pitch_stopvelmax
      if (PID_pit_var.velmax.lt.pitch_stopvelmax2) PID_pit_var.velmax=pitch_stopvelmax2
    case default
      write(6,'(a,i2,a)') ' *** ERROR *** Stop type ',pitch_stoptype,' not known'
      stop
  end select
endif
!-------------------------------------------------------------------------------------------------
! Compute PID feedback to generator torque
!-------------------------------------------------------------------------------------------------
if (DT_mode_filt.f0.gt.0.d0) then
  e_pitch(1)=notch2orderfilt(deltat,stepno,DT_mode_filt,omega_err_filt_pitch)
  e_pitch(2)=notch2orderfilt(deltat,stepno,pwr_DT_mode_filt,Pe_ref-Pe_rated)
else
  e_pitch(1)=omega_err_filt_pitch
  e_pitch(2)=Pe_ref-Pe_rated
endif
theta_col_ref=PID2(stepno,deltat,kgain_pitch,PID_pit_var,e_pitch)
thetaref=theta_col_ref
!*************************************************************************************************
! Output
!*************************************************************************************************
array2( 1)=Qgen_ref             !    1: Generator torque reference            [Nm]
array2( 2)=thetaref(1)          !    2: Pitch angle reference of blade 1      [rad]
array2( 3)=thetaref(2)          !    3: Pitch angle reference of blade 2      [rad]
array2( 4)=thetaref(3)          !    4: Pitch angle reference of blade 3      [rad]
array2( 5)=Pe_ref               !    5: Power reference                       [W]
array2( 6)=WSPfilt              !    6: Filtered wind speed                   [m/s]
array2( 7)=omegafilt            !    7: Filtered rotor speed                  [rad/s]
array2( 8)=omega_err_filt_speed !    8: Filtered rotor speed error for torque [rad/s]
array2( 9)=omega_dtfilt         !    9: Bandpass filtered rotor speed         [rad/s]
array2(10)=PID_gen_var.outpro   !   10: Proportional term of torque contr.    [Nm]
array2(11)=PID_gen_var.outset   !   11: Integral term of torque controller    [Nm]
array2(12)=PID_gen_var.outmin   !   12: Minimum limit of torque               [Nm]
array2(13)=PID_gen_var.outmax   !   13: Maximum limit of torque               [Nm]
array2(14)=switch1              !   14: Torque limit switch based on pitch    [-]
array2(15)=omega_err_filt_pitch !   15: Filtered rotor speed error for pitch  [rad/s]
array2(16)=e_pitch(2)           !   16: Power error for pitch                 [W]
array2(17)=PID_pit_var.outpro   !   17: Proportional term of pitch controller [rad]
array2(18)=PID_pit_var.outset   !   18: Integral term of pitch controller     [rad]
array2(19)=PID_pit_var.outmin   !   19: Minimum limit of pitch                [rad]
array2(20)=PID_pit_var.outmax   !   20: Maximum limit of pitch                [rad]
array2(21)=Qdamp_ref            !   21: Torque reference from DT damper       [Nm]
return
end subroutine update_regulation
!*************************************************************************************************
```

A.2. Functional routines in risoe_controller_fcns.f90

```
module risoe_controller_fcns
! Constants
real*8 pi,degrad,raddeg
parameter(pi=3.14159265358979,degrad=0.0174532925,raddeg=57.2957795131)
integer*4 maxwplines
parameter(maxwplines=100)
! Types
type Tfirstordervar
  real*8 tau,x1,x1_old,y1,y1_old
  integer*4 stepno1
end type Tfirstordervar
type Tlowpass2order
  real*8 zeta,f0,x1,x2,x1_old,x2_old,y1,y2,y1_old,y2_old
  integer*4 stepno1
end type Tlowpass2order
type Tnotch2order
  real*8::zeta1=0.1
  real*8::zeta2=0.001
  real*8 f0,x1,x2,x1_old,x2_old,y1,y2,y1_old,y2_old
  integer*4 stepno1
end type Tnotch2order
type Tbandpassfilt
  real*8::zeta=0.02
  real*8::tau=0.0
  real*8 f0,x1,x2,x1_old,x2_old,y1,y2,y1_old,y2_old
  integer*4 stepno1
end type Tbandpassfilt
type Tpidvar
  real*8 Kpro,Kdif,Kint,outmin,outmax,velmax,error1,outset1,outres1
  integer*4 stepno1
  real*8 outset,outpro,outdif,error1_old,outset1_old,outres1_old,outres
end type Tpidvar
type Tpid2var
  real*8 Kpro(2),Kdif(2),Kint(2),outmin,outmax,velmax,error1(2),outset1,outres1
  integer*4 stepno1
  real*8 outset,outpro,outdif,error1_old(2),outset1_old,outres1_old,outres
end type Tpid2var
type Twpdata
  real*8 wpdata(maxwplines,2)
  integer*4 lines
end type Twpdata
! Variables
integer*4 stepno
logical const_power
real*8 deltat,time_old
real*8 omega_ref_max,omega_ref_min,Pe_rated,Qg_rated,pitch_stopang,max_lss_torque
real*8 Kopt,rel_sp_open_Qg
real*8 kk1,kk2,rel_limit
real*8 switch1_pitang_lower,switch1_pitang_upper,switch1
real*8 DT_damp_gain
logical::generator_cutin=.false.
real*8 t_cutin,t_generator_cutin,t_cutin_delay
integer*4 pitch_stoptype
real*8 t_cutout,pitch_stopdelay,pitch_stopdelay2,pitch_stopvelmax,pitch_stopvelmax2
type(Tlowpass2order) omega2ordervar
type(Tnotch2order) DT_mode_filt
type(Tnotch2order) pwr_DT_mode_filt
type(Tbandpassfilt) DT_damper_filt
type(Tpid2var) PID_pit_var
type(Tpidvar) PID_gen_var
type(Twpdata) OPdatavar
type(Tfirstordervar) wspfirstordervar
type(Tfirstordervar) pitchfirstordervar
```

```
type(Tfirstordervar) torquefirstordervar
type(Tfirstordervar) switchfirstordervar
type(Tfirstordervar) cutinfirstordervar
!***************************************************************************************************
contains
!***************************************************************************************************
function switch_spline(x,x0,x1)
implicit none
! A function that goes from 0 at x0 to 1 at x1
real*8 switch_spline,x,x0,x1
if (x0.ge.x1) then
  if (x.lt.x0) then
    switch_spline=0.d0
  else
    switch_spline=1.d0
  endif
elseif (x0.gt.x1) then
  switch_spline=0.d0
else
  if (x.lt.x0) then
    switch_spline=0.d0
  elseif (x.gt.x1) then
    switch_spline=1.d0
  else
    switch_spline=2.d0/(-x1+x0)**3*x**3+(-3.d0*x0-3.d0*x1)/(-x1+x0)**3*x**2&
                  +6.d0*x1*x0/(-x1+x0)**3*x+(x0-3.d0*x1)*x0**2/(-x1+x0)**3
  endif
endif
return
end function switch_spline
!***************************************************************************************************
function interpolate(x,x0,x1,f0,f1)
implicit none
real*8 interpolate,x,x0,x1,f0,f1
if (x0.eq.x1) then
  interpolate=f0
else
  interpolate=(x-x1)/(x0-x1)*f0+(x-x0)/(x1-x0)*f1
endif
return
end function interpolate
!***************************************************************************************************
function GetOptiPitch(wsp)
implicit none
real*8 GetOptiPitch,wsp
! local vars
real*8 x,x0,x1,f0,f1,pitch
integer*4 i
i=1
do while((OPdatavar.wpdata(i,1).le.wsp).and.(i.le.OPdatavar.lines))
  i=i+1
enddo
if (i.eq.1) then
  GetOptiPitch=OPdatavar.wpdata(1,2)
elseif (i.gt.OPdatavar.lines) then
  GetOptiPitch=OPdatavar.wpdata(OPdatavar.lines,2)
else
  x=wsp
  x0=OPdatavar.wpdata(i-1,1)
  x1=OPdatavar.wpdata(i,1)
  f0=OPdatavar.wpdata(i-1,2)
  f1=OPdatavar.wpdata(i,2)
  Pitch=interpolate(x,x0,x1,f0,f1)
  GetOptiPitch=Pitch
endif
return
```

```
end function GetOptiPitch
!***************************************************************************************************
function lowpass1orderfilt(dt,stepno,filt,x)
implicit none
integer*4 stepno
real*8 lowpass1orderfilt,dt,x,y,a1,b1,b0,tau
type(Tfirstordervar) filt
! Step
if ((stepno.eq.1).and.(stepno.gt.filt.stepno1)) then
  filt.x1_old=x
  filt.y1_old=x
  y=x
else
  if (stepno.gt.filt.stepno1) then
    filt.x1_old=filt.x1
    filt.y1_old=filt.y1
  endif
  tau=filt.tau
  a1 = (2 * tau - dt) / (2 * tau + dt)
  b0 = dt / (2 * tau + dt)
  b1 = b0
  y=a1*filt.y1_old+b0*x+b1*filt.x1_old
endif
! Save previous values
filt.x1=x
filt.y1=y
filt.stepno1=stepno
! Output
lowpass1orderfilt=y
return
end function lowpass1orderfilt
!***************************************************************************************************
function lowpass2orderfilt(dt,stepno,filt,x)
implicit none
real*8 lowpass2orderfilt(2),dt,x
integer*4 stepno
type(Tlowpass2order) filt
! local vars
real*8 y,f0,zeta,a1,a2,b0,b1,b2,denom
! Step
if ((stepno.eq.1).and.(stepno.gt.filt.stepno1)) then
  filt.x1=x
  filt.x2=x
  filt.x1_old=filt.x1
  filt.x2_old=filt.x2
  filt.y1=x
  filt.y2=x
  filt.y1_old=filt.y1
  filt.y2_old=filt.y2
  y=x
else
  if (stepno.gt.filt.stepno1) then
    filt.x1_old=filt.x1
    filt.x2_old=filt.x2
    filt.y1_old=filt.y1
    filt.y2_old=filt.y2
  endif
  f0=filt.f0
  zeta=filt.zeta
  denom=3.d0+6.d0*zeta*pi*f0*dt+4.d0*pi**2*f0**2*dt**2
  a1=(6.d0-4.d0*pi**2*f0**2*dt**2)/denom
  a2=(-3.d0+6.d0*zeta*pi*f0*dt-4.d0*pi**2*f0**2*dt**2)/denom
  b0=4.d0*pi**2*f0**2*dt**2/denom
  b1=b0
  b2=b0
  y=a1*filt.y1_old+a2*filt.y2_old+b0*x+b1*filt.x1_old+b2*filt.x2_old
```

```
endif
! Save previous values
filt.x2=filt.x1
filt.x1=x
filt.y2=filt.y1
filt.y1=y
filt.stepno1=stepno
! Output
lowpass2orderfilt(1)=y
lowpass2orderfilt(2)=0.5d0*(y-filt.y2_old)/dt
return
end function lowpass2orderfilt
!*****************************************************************************************************
function notch2orderfilt(dt,stepno,filt,x)
implicit none
real*8 notch2orderfilt,dt,x
integer*4 stepno
type(Tnotch2order) filt
! local vars
real*8 y,f0,zeta1,zeta2,a1,a2,b0,b1,b2,denom
! Step
if ((stepno.eq.1).and.(stepno.gt.filt.stepno1)) then
  filt.x1=x
  filt.x2=x
  filt.x1_old=filt.x1
  filt.x2_old=filt.x2
  filt.y1=x
  filt.y2=x
  filt.y1_old=filt.y1
  filt.y2_old=filt.y2
  y=x
else
  if (stepno.gt.filt.stepno1) then
    filt.x1_old=filt.x1
    filt.x2_old=filt.x2
    filt.y1_old=filt.y1
    filt.y2_old=filt.y2
  endif
  f0=filt.f0
  zeta1=filt.zeta1
  zeta2=filt.zeta2
  denom=3.d0+6.d0*zeta1*pi*f0*dt+4.d0*pi**2*f0**2*dt**2
  a1=(6.d0-4.d0*pi**2*f0**2*dt**2)/denom
  a2=(-3.d0+6.d0*zeta1*pi*f0*dt-4.d0*pi**2*f0**2*dt**2)/denom
  b0=(3.d0+6.d0*zeta2*pi*f0*dt+4.d0*pi**2*f0**2*dt**2)/denom
  b1=(-6.d0+4.d0*pi**2*f0**2*dt**2)/denom
  b2=(3.d0-6.d0*zeta2*pi*f0*dt+4.d0*pi**2*f0**2*dt**2)/denom
  y=a1*filt.y1_old+a2*filt.y2_old+b0*x+b1*filt.x1_old+b2*filt.x2_old
endif
! Save previous values
filt.x2=filt.x1
filt.x1=x
filt.y2=filt.y1
filt.y1=y
filt.stepno1=stepno
! Output
notch2orderfilt=y
return
end function notch2orderfilt
!*****************************************************************************************************
function bandpassfilt(dt,stepno,filt,x)
implicit none
real*8 bandpassfilt,dt,x
integer*4 stepno
type(Tbandpassfilt) filt
! local vars
```

```
real*8 y,f0,zeta,tau,a1,a2,b0,b1,b2,denom
! Step
if ((stepno.eq.1).and.(stepno.gt.filt.stepno1)) then
  filt.x1=x
  filt.x2=x
  filt.x1_old=filt.x1
  filt.x2_old=filt.x2
  filt.y1=x
  filt.y2=x
  filt.y1_old=filt.y1
  filt.y2_old=filt.y2
  y=x
else
  if (stepno.gt.filt.stepno1) then
    filt.x1_old=filt.x1
    filt.x2_old=filt.x2
    filt.y1_old=filt.y1
    filt.y2_old=filt.y2
  endif
  f0=filt.f0
  zeta=filt.zeta
  tau=filt.tau
  denom=3.d0+6.d0*zeta*pi*f0*dt+4.d0*pi**2*f0**2*dt**2
  a1=-(-6.d0+4.d0*pi**2*f0**2*dt**2)/denom
  a2=-(3.d0-6.d0*zeta*pi*f0*dt+4.d0*pi**2*f0**2*dt**2)/denom
  b0=-(-6.d0*zeta*pi*f0*dt-12.d0*zeta*pi*f0*tau)/denom
  b1=-24.d0*zeta*pi*f0*tau/denom
  b2=-(6.d0*zeta*pi*f0*dt-12.d0*zeta*pi*f0*tau)/denom
  y=a1*filt.y1_old+a2*filt.y2_old+b0*x+b1*filt.x1_old+b2*filt.x2_old
endif
! Save previous values
filt.x2=filt.x1
filt.x1=x
filt.y2=filt.y1
filt.y1=y
filt.stepno1=stepno
! Output
bandpassfilt=y
return
end function bandpassfilt
!*****************************************************************************************************
function PID(stepno,dt,kgain,PIDvar,error)
implicit none
integer*4 stepno
real*8 PID,dt,kgain(3),error
type(Tpidvar) PIDvar
! Local vars
real*8 eps
parameter(eps=1.d-6)
! Initiate
if (stepno.eq.1) then
  PIDvar.outset1=0
  PIDvar.outres1=0
  PIDvar.error1=0
  PIDvar.error1_old=0.0
  PIDvar.outset1_old=0.0
  PIDvar.outres1_old=0.0
endif
! Save previous values
if (stepno.gt.PIDvar.stepno1) then
  PIDvar.outset1_old=PIDvar.outset1
  PIDvar.outres1_old=PIDvar.outres1
  PIDvar.error1_old=PIDvar.error1
endif
! Update the integral term
PIDvar.outset=PIDvar.outset1_old+0.5d0*(error+PIDvar.error1)*Kgain(2)*PIDvar.Kint*dt
```

```
! Update proportional term
PIDvar.outpro=Kgain(1)*PIDvar.Kpro*0.5d0*(error+PIDvar.error1)
! Update differential term
PIDvar.outdif=Kgain(3)*PIDvar.Kdif*(error-PIDvar.error1_old)/dt
! Sum to up
PIDvar.outres=PIDvar.outset+PIDvar.outpro+PIDvar.outdif
! Satisfy hard limits
if (PIDvar.outres.lt.PIDvar.outmin) then
  PIDvar.outres=PIDvar.outmin
elseif (PIDvar.outres.gt.PIDvar.outmax) then
  PIDvar.outres=PIDvar.outmax
endif
! Satisfy max velocity
if (PIDvar.velmax.gt.eps) then
    if ((abs(PIDvar.outres-PIDvar.outres1_old)/dt).gt.PIDvar.velmax) &
      PIDvar.outres=PIDvar.outres1_old+dsign(PIDvar.velmax*dt,PIDvar.outres-PIDvar.outres1_old)
endif
! Anti-windup on integral term and save results
PIDvar.outset1=PIDvar.outres-PIDvar.outpro-PIDvar.outdif
PIDvar.outres1=PIDvar.outres
PIDvar.error1=error
PIDvar.stepno1=stepno
! Set output
if (stepno.eq.0) then
  PID=0
else
  PID=PIDvar.outres
endif
return
end function PID
!***************************************************************************************************
function PID2(stepno,dt,kgain,PIDvar,error)
implicit none
integer*4 stepno
real*8 PID2,dt,kgain(3,2),error(2)
type(Tpid2var) PIDvar
! Local vars
real*8 eps
parameter(eps=1.d-6)
! Initiate
if (stepno.eq.1) then
  PIDvar.outset1=0
  PIDvar.outres1=0
  PIDvar.error1=0
  PIDvar.error1_old=0.0
  PIDvar.outset1_old=0.0
  PIDvar.outres1_old=0.0
endif
! Save previous values
if (stepno.gt.PIDvar.stepno1) then
  PIDvar.outset1_old=PIDvar.outset1
  PIDvar.outres1_old=PIDvar.outres1
  PIDvar.error1_old=PIDvar.error1
endif
! Update the integral term
PIDvar.outset=PIDvar.outset1_old+0.5d0*dt*(Kgain(2,1)*PIDvar.Kint(1)*(error(1)+PIDvar.error1(1))&
                                          +Kgain(2,2)*PIDvar.Kint(2)*(error(2)+PIDvar.error1(2)))
! Update proportional term
PIDvar.outpro=0.5d0*(Kgain(1,1)*PIDvar.Kpro(1)*(error(1)+PIDvar.error1(1))&
                    +Kgain(1,2)*PIDvar.Kpro(2)*(error(2)+PIDvar.error1(2)))
! Update differential term
PIDvar.outdif=(Kgain(3,1)*PIDvar.Kdif(1)*(error(1)-PIDvar.error1_old(1)))/dt
! Sum to up
PIDvar.outres=PIDvar.outset+PIDvar.outpro+PIDvar.outdif
! Satisfy hard limits
if (PIDvar.outres.lt.PIDvar.outmin) then
```

```
  PIDvar.outres=PIDvar.outmin
elseif (PIDvar.outres.gt.PIDvar.outmax) then
  PIDvar.outres=PIDvar.outmax
endif
! Satisfy max velocity
if (PIDvar.velmax.gt.eps) then
    if ((abs(PIDvar.outres-PIDvar.outres1_old)/dt).gt.PIDvar.velmax) &
      PIDvar.outres=PIDvar.outres1_old+dsign(PIDvar.velmax*dt,PIDvar.outres-PIDvar.outres1_old)
endif
! Anti-windup on integral term and save results
PIDvar.outset1=PIDvar.outres-PIDvar.outpro-PIDvar.outdif
PIDvar.outres1=PIDvar.outres
PIDvar.error1=error
PIDvar.stepno1=stepno
! Set output
if (stepno.eq.0) then
  PID2=0
else
  PID2=PIDvar.outres
endif
return
end function PID2
!***************************************************************************************************
end module risoe_controller_fcns
```

Appendix B.

Discrete filters

This appendix contains the derivations of the discrete filters and a test of their validity. Some of the parameters of the filters are hardcoded in the current implementation as shown in the previous appendix. The values of these hardcoded parameters are repeated herein.

B.1. First order filter

In continuous form the first order low-pass filter can be written as

$$\dot{\bar{x}} + \tau\bar{x} = \tau x \tag{B.1}$$

where $x = x(t)$ is the original signal, $\bar{x}(t)$ is the filtered signal, τ is the user-defined time constant of the filter, and $\dot{(\,)} = /dt$ denotes the time derivative. The discrete first order low-pass filter used in the controller is derived from this continuous formulation by approximating the states and their time derivatives as an average based on the previous and current step:

$$x(t) \approx \frac{x_k + x_{k-1}}{2} \quad \text{and} \quad \dot{x}(t) \approx \frac{x_k - x_{k-1}}{\Delta t} \tag{B.2}$$

where k is the index of the current time step, and Δt is the time step length. Substitution into (B.1) and rearranging the terms, the f_1-function is obtained as

$$f_1\left(\tau; \bar{x}_{k-1}, x_k, x_{k-1}\right) = a_1\bar{x}_{k-1} + b_0 x_k + b_1 x_{k-1} \tag{B.3}$$

where

$$a_1 = \frac{2\tau - \Delta t}{2\tau + \Delta t}, \quad b_0 = \frac{\Delta t}{2\tau + \Delta t}, \quad b_1 = b_0 \tag{B.4}$$

To test this discrete filter, the Finite-Difference Method for constructing periodic solutions [3] has been used to obtain the period solutions to harmonic excitation of the filter with a large number of different excitation frequencies. Figure B.1 shows the amplitude and phase of these solutions (red circles) compared to the transfer function obtained by transformation of the continuous formulation (B.1) into the frequency domain.

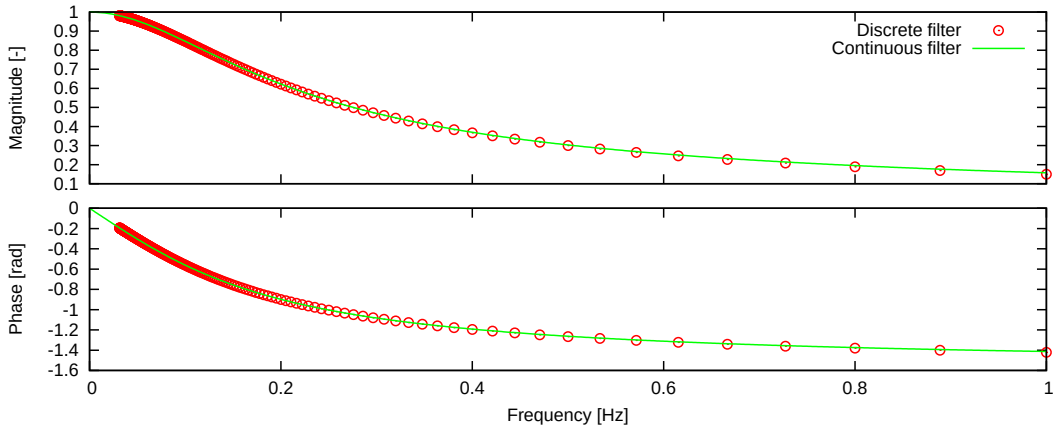

Figure B.1.: Test of the first order low-pass filter function (B.3) with a time constant of $\tau = 1$ s and a time step of $\Delta t = 0.125$ s.

B.2. Second order filters

The discrete second filters used in the controller are derived from the continuous formulations by approximating the states and their time derivatives as an average based on the two previous and current step:

$$x(t) \approx \frac{x_k + x_{k-1} + x_{k-2}}{3}, \quad \dot{x}(t) \approx \frac{x_k - x_{k-2}}{2\Delta t} \quad \text{and} \quad \ddot{x}(t) \approx \frac{x_k - 2x_{k-1} + x_{k-2}}{\Delta t^2} \tag{B.5}$$

The continuous form of the second order low-pass filter is

$$\ddot{\bar{x}} + 2\zeta\omega\dot{\bar{x}} + \omega^2\bar{x} = \omega^2 x \tag{B.6}$$

where again $x = x(t)$ and $\bar{x}(t)$ are the original and filtered signals, respectively. The parameters ζ and ω are the user-defined damping ratio and frequency of the filter. Substitution of (B.5) into this equations yields the discrete second order low-pass filter function:

$$f_2\left(\zeta, \omega; \bar{x}_{k-1}, \bar{x}_{k-2}, x_k, x_{k-1}, x_{k-2}\right) = a_1\bar{x}_{k-1} + a_2\bar{x}_{k-2} + b_0x_k + b_1x_{k-1} + b_2x_{k-2} \tag{B.7}$$

where

$$a_1 = \frac{6 - \omega^2\Delta t^2}{d}, \quad a_2 = \frac{-3 + 3\,\zeta\,\omega\,\Delta t - \omega^2\Delta t^2}{d},$$
$$b_0 = \frac{\omega^2\Delta t^2}{d}, \quad b_1 = b_0 \quad b_2 = b_0 \tag{B.8}$$

where the common denominator is $d = 3 + 3\zeta\omega\Delta t + \omega^2\Delta t^2$.

The continuous form of the second order notch filter is

$$\ddot{\bar{x}} + 2\zeta_1\omega\dot{\bar{x}} + \omega^2\bar{x} = \ddot{x} + 2\zeta_2\omega\dot{x} + \omega^2 x \tag{B.9}$$

where $\zeta_1 = 0.1$, $\zeta_2 = 0.001$ and ω are the hardcoded damping ratios and the user-defined frequency of the notch filter, respectively. Substitution of (B.5) into this equations yields the discrete second order notch filter function:

$$f_n\left(\zeta_1, \zeta_2, \omega; \bar{x}_{k-1}, \bar{x}_{k-2}, x_k, x_{k-1}, x_{k-2}\right) = a_1\bar{x}_{k-1} + a_2\bar{x}_{k-2} + b_0x_k + b_1x_{k-1} + b_2x_{k-2} \tag{B.10}$$

where

$$a_1 = -\frac{-6+\omega^2\Delta t^2}{d}, \quad a_2 = -\frac{3-3\,\zeta_1\,\omega\,\Delta t+\omega^2\Delta t^2}{d},$$
$$b_0 = \frac{3+3\,\zeta_2\,\omega\,\Delta t+\omega^2\Delta t^2}{d}, \quad b_1 = -a_1, \quad b_2 = \frac{3-3\,\zeta_2\,\omega\,\Delta t+\omega^2\Delta t^2}{d} \tag{B.11}$$

where the common denominator is $d = 3+3\,\zeta_1\,\omega\,\Delta t+\omega^2\Delta t^2$.

The continuous form of the second order band-pass filter used in the controller is

$$\dddot{\bar{x}} + 2\zeta\omega\ddot{\bar{x}} + \omega^2\dot{\bar{x}} = 2\zeta\omega\left(\ddot{x} + \tau\dddot{x}\right) \tag{B.12}$$

where $\zeta = 0.02$ and ω are the hardcoded damping ratio and the user-defined frequency of the band-pass filter, respectively. The additional parameter τ is a time constant which in the current implementation is hardcoded to zero. Substitution of (B.5) into this equations yields the discrete second order band-pass filter function:

$$f_p\left(\zeta,\omega;\bar{x}_{k-1},\bar{x}_{k-2},x_k,x_{k-1},x_{k-2}\right) = a_1\bar{x}_{k-1} + a_2\bar{x}_{k-2} + b_0x_k + b_1x_{k-1} + b_2x_{k-2} \tag{B.13}$$

where

$$a_1 = -\frac{-6+\omega^2\Delta t^2}{d}, \quad a_2 = -\frac{3-3\,\zeta\,\omega\,\Delta t+\omega^2\Delta t^2}{d},$$
$$b_0 = 3\,\frac{\zeta\,\omega\,\left(\Delta t+2\,\tau\right)}{d}, \quad b_1 = -12\,\frac{\zeta\,\omega\,\tau}{d}, \quad b_2 = -3\,\frac{\zeta\,\omega\,\left(\Delta t-2\,\tau\right)}{d} \tag{B.14}$$

where the common denominator is $d = 3+3\zeta\omega\Delta t+\omega^2\Delta t^2$. Notice that the hardcoded parameter τ is not included in the list of parameters in the function call.

Test and validation of these second order filters are shown in Figures B.2 – B.4.

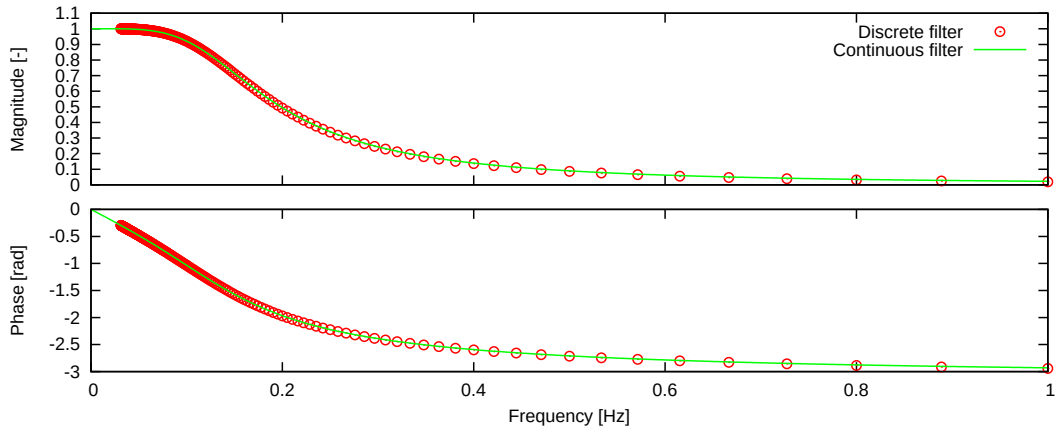

Figure B.2.: Test of the second order low-pass filter function (B.7) with a damping ratio of $\zeta = 0.7$, a frequency of $\omega = 0.15$ Hz and a time step of $\Delta t = 0.125$ s.

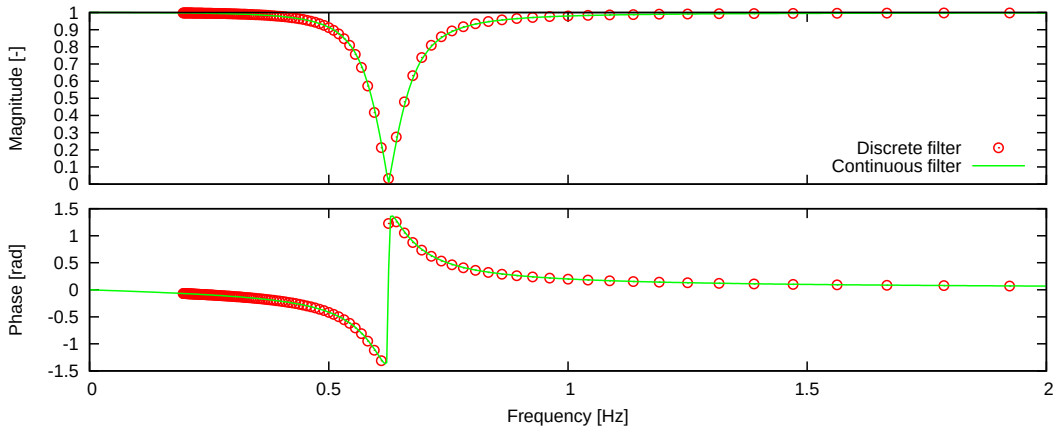

Figure B.3.: Test of the second order notch filter function (B.10) with a frequency of $\omega = 0.625$ Hz and a time step of $\Delta t = 0.04$ s.

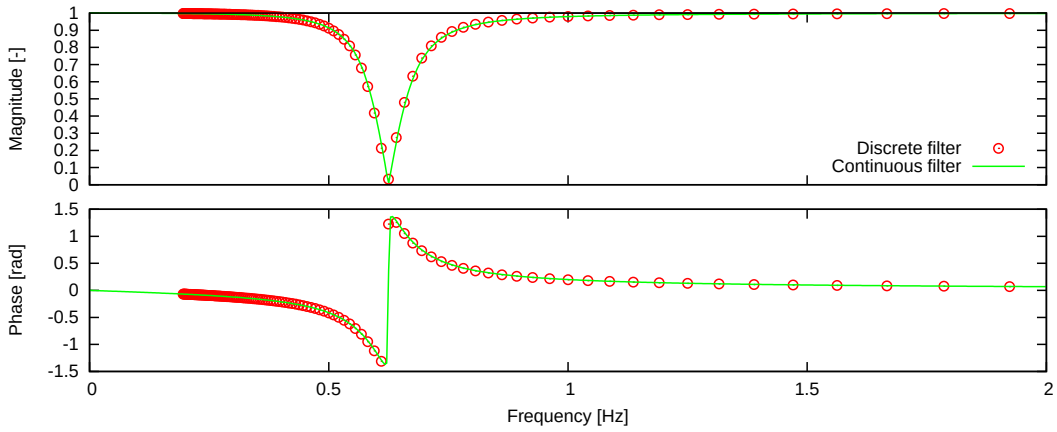

Figure B.4.: Test of the second order band-pass filter function (B.13) with a frequency of $\omega = 0.625$ Hz and a time step of $\Delta t = 0.04$ s.

DTU Wind Energy

DTU Wind Energy is a department of the Technical University of Denmark with a unique integration of research, education, innovation and public/private sector consulting in the field of wind energy. Our activities develop new opportunities and technology for the global and Danish exploitation of wind energy. Research focuses on key technical-scientific fields, which are central for the development, innovation and use of wind energy and provides the basis for advanced education at the education.

We have more than 230 staff members of which approximately 60 are PhD students. Research is conducted within 9 research programmes organized into three main topics: Wind energy systems, Wind turbine technology and Basics for wind energy.

Technical University of Denmark
DTU Vindenergi
Frederiksborgvej 399
Bygning 118
4000 Roskilde
Telefon 46 77 50 85
info@vindenergi.dtu.dk
www.vindenergi.dtu.dk